AD/A-004 357

WORKING PAPERS IN SPEECH RECOGNITION, III

D. Raj Reddy

Carnegie-Mellon University

Prepared for:

Air Force Office of Scientific Research
Advanced Research Projects Agency

April 1974

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR - TR - 75 - 0135	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD/A-004357
4. TITLE (and Subtitle) WORKING PAPERS IN SPEECH RECOGNITION III		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) CMU Computer Science Speech Group Headed by D. Raj Reddy		8. CONTRACT OR GRANT NUMBER(s) F44620-73-C-0074
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Department of Computer Science Pittsburgh, Pennsylvania 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61101D AO 2466
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE April 1974
		13. NUMBER OF PAGES 47
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Air Force Office of Scientific Research (NM) 1400 Wilson Boulevard Arlington, Virginia 22209		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is a collection of Working Papers in Speech Recognition by the following:
V. R. Lesser, R. D. Fennell, L. D. Erman, and D. R. Reddy, Organization of the HEARSAY II Speech Understanding System".
James Baker, "The DRAGON System -- An Overview".
H. G. Goldberg, D. R. Reddy, and R. L. Suslick, "Parameter-Independent Machine Segmentation and Labeling".
Janet Baker, "ANew Time-domain Analysis of Fricatives and Stop Consonants".

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

20.(Abstract Continued)

L. Shockey and L. D. Erman, "Sub-Lexical Levels in the HEARSAY II Speech Understanding System".
E. Rich, "Inference and Use of Simple Predictive Grammars".
M. J. Knudsen, "Real-time Linear-predictive Coding of Speech on the SPS-41 Microprogrammed Triple-processor System".
S. Kriz, "A 16-bit A-D-A Conversion System for High-Fidelity Audio Research".

Working Papers In Speech Recognition

- III -

CMU Computer Science Speech Group
April, 1974

This report reprints the CMU contributed papers from the IEEE Symposium on Speech Recognition, April 15-19, 1974, held at CMU. The paper numbers and page numbers are consistent with the Symposium Contributed Papers, IEEE catalog no. 74CH0878-9 AE, copyright 1974 by The Institute of Electrical and Electronics Engineers, Inc. Copies of the Contributed Papers are available from IEEE, 345 E. 47 St., N.Y., N.Y. 10017.

M2 V. R. Lesser, R. D. Fennell, L. D. Erman, & D. R. Reddy, "Organization of the HEARSAY II Speech Understanding System.

M3 James Baker, "The DRAGON System -- An Overview".

T7 H. G. Goldberg, D. R. Reddy, & R. L. Suslick, "Parameter-Independent Machine Segmentation and Labeling".

T11 Janet Baker, "A New Time-domain Analysis of Fricatives and Stop Consonants."

W13 L. Shockey & L. D. Erman, "Sub-Lexical Levels in the HEARSAY II Speech Understanding System".

H5 E. Rich, "Inference and Use of Simple Predictive Grammars".

F5 M. J. Knudsen, "Real-time Linear-predictive Coding of Speech on the SPS-41 Microprogrammed Triple-processor System".

F6 S. Kriz, "A 16-bit A-D-A Conversion System for High-Fidelity Audio Research".

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

ORGANIZATION OF THE HEARSAY II SPEECH UNDERSTANDING SYSTEM

Victor R. Lesser, Richard D. Fennell, Lee D. Erman, and D. Raj Reddy

Computer Science Department*
Carnegie-Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

Hearsay II (HSII) is a system currently under development at Carnegie-Mellon University to study the connected speech understanding problem. It is similar to Hearsay I (HSI) in that it is based on the hypothesize-and-test paradigm, using cooperating independent knowledge sources communicating with each other through a global data structure (blackboard). It differs in the sense that many of current limitations and shortcomings of HSI are resolved in HSII.

The main new features of the Hearsay II system structure are: 1) the representation of knowledge as self-activating, asynchronous, parallel processes, 2) the representation of the partial analysis in a generalized 3-dimensional network (the dimensions being level of representation (e.g., acoustic, phonetic, phonemic, lexical, syntactic), time, and alternatives) with contextual and structural support connections explicitly specified, 3) a convenient modular structure for incorporating new knowledge into the system at any level, and 4) a system structure suitable for execution on a parallel processing system.

The main task domain under study is the retrieval of daily wire-service news stories upon voice request by the user. The main parametric representations used for this study are 1/3-octave filter-bank and LPC-derived vocal tract parameters (Knudsen, 1974, and Kriz, 1974). The acoustic segmentation and labeling procedures are parameter-independent (Goldberg, et al., 1974). The acoustic, phonetic, and phonological components (Shockey and Erman, 1974) are feature-based rewriting rules which transform the segmental units into higher-level phonetic units. The vocabulary size for the task is approximately 1200 words. This vocabulary information is used to generate word-level hypotheses from phonetic and surface-phonemic levels based on prosodic (stress) information. The syntax for the task permits simple English-like sentences and is used to generate hypotheses based on the probability of occurrence of that grammatical construct (Rich, 1974). The semantic model is based on the news items of the day, analysis of the conversation, and the presence of certain content words in the partial analysis. This knowledge is to be represented as a production system. The system is expected to be operational on a 16-processor mini-computer system (Bell, et al., 1971) being built at CMU.

This paper deals primarily with the issues of the system organization of the Hearsay II system.

INTRODUCTION

The Hearsay II (HSII) speech understanding system is a successor to the Hearsay (HSI) system (Reddy, et al., 1973a, 1973b). HSII represents, in terms of both its system organization and its speech knowledge, a significant increase in sophistication and generality over HSI. The development of HSII has been based on two years of experience with a running version of HSI, a desire to exploit multiprocessor and network computer architecture for efficient implementation (Bell, et al., 1971, 1973, and Erman, et al., 1973), and a desire to handle more complex speech task domains (e.g., larger vocabularies, less restricted grammars, and a more complete set of knowledge sources including prosodics, user models, etc.). While from a conceptual point of view HSII is a natural extension of the framework that HSI posited for a speech understanding system, it differs significantly in its design and in its details of implementation.

The HEARSAY System Model

HSI was based on the view that the inherently errorful nature of connected speech processing could be handled only through the efficient use of multiple, diverse sources of knowledge (Reddy, et al., 1970, and Newell, et al., 1971). The major focus of the design of HSI was the development of a framework for representing these diverse sources of knowledge and their cooperation (Reddy and Newell, 1974). This framework is the conceptual legacy which forms the basis for the HSII design.

There are four dimensions along which knowledge representation in HSI can be described:

1) function,
2) structure,
3) cooperation,
4) attention focusing.

The <u>function</u> of a knowledge source (KS) in HSI has three aspects. The first is for the KS to know when it has something useful to contribute, the second is to contribute its knowledge through the mechanism of making a **hypothesis** (guess) about some aspect of the speech utterance, and the third is to evaluate the contribution of other knowledge sources, i.e., to verify and reorder (or reject) the hypotheses made by other knowledge sources. Each of these aspects of a KS is carried out with respect to a particular context, the context being some subset of the previously generated hypotheses. Thus, new knowledge is built upon the educated guesses made at some previous time by other knowledge sources.

The <u>structure</u> of each knowledge source in HSI is specified so that it is independent and separable from all other KS's in the system. This permits the easy addition of new types of KS's and replacement of KS's with alternative versions of those KS's. Thus, the system structure can be easily adapted to new speech task domains which have KS's specific to that domain, and the contribution of a particular KS to the total recognition effort can be more easily evaluated.

* This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

The choice of a framework for <u>cooperation</u> among knowledge sources is intimately interwoven with the function and structure of knowledge in HSI. The mechanism for KS cooperation involves **hypothesizing** and **testing** (creating and evaluating) hypotheses in a **global data base** (blackboard). The generation and modification of globally accessible hypotheses thus becomes the primary means of communication between diverse KS's. This mechanism of cooperation allows a KS to contribute knowledge without being aware of which other KS's will use its knowledge or which KS contributed the knowledge that it used. Thus, each KS can be made independent and separable.

The global data base that KS's use for cooperation contains many possible interpretations of the speech data. Each of these interpretations represents a "limited" context in which a KS can possibly contribute information by proposing or validating hypotheses. **Attention focusing** of a KS involves choosing which of these limited contexts it will operate in and for how much processing time. The attention focusing strategy is decoupled from the functions of individual knowledge sources. Thus, the decision of whether a KS can contribute in a particular context is local to the KS, while the assignment of that KS to one of the many contexts on which it can possibly operate is made more globally. This decoupling of focusing strategy from knowledge acquisition, together with the decoupling of the data environment (global data base) from control flow (KS invocation) and the limited context in which a KS operates, permits a quick refocusing of attention of KS's. The ability to refocus quickly is very important in a speech understanding system because the errorful nature of speech data and processing leads to many potential interpretations of the speech. Thus, as soon as possible after an interpretation no longer seems the most promising, the activity of the system should be refocused to the new most promising interpretation.

OVERVIEW OF HEARSAY I

The following is a brief description of the HSI implementation for this model of knowledge source representation and cooperation. (A more complete description is contained in Reddy, et al., 1973a, 1973b.) This description will then be used to contrast the differences of implementation philosophy between HSI and HSII.

HEARSAY I Implementation Overview

The global data base of HSI consists of partial sentence hypotheses, each being a sequence of words with non-overlapping time locations in the utterance. It is a <u>partial</u> sentence hypothesis because not all of the utterance may be described by the given sequence of words. In particular, gaps in the knowledge of the utterance are designated by "filler" words. The partial sentence hypotheses also contain confidence ratings for each word hypothesis and a composite rating for the overall sequence of words. A sentence hypothesis is the focal point that is used to invoke a knowledge source. The sentence hypothesis also contains the accumulation of all information that any knowledge source has contributed to that hypothesis.

Knowledge sources are invoked in a lockstep sequence consisting of three phases: poll, **hypothesize**, and **test**. At each phase, all knowledge sources are invoked for that phase, and the next phase does not commence until all KS's have completed the current one. The poll phase involves determining which KS's have something to contribute to the sentence hypothesis which is currently being focused upon; polling also determines how confident each KS is about its proposed contributions. The hypothesize phase consists of invoking the KS showing the most confidence about its proposed contribution of knowledge. This KS then hypothesizes a set of possible words (option words) for some (one) "filler" word in the speech utterance. The testing phase consists of each KS evaluating (verifying) the possible option words with respect to the given context. After all knowledge sources have completed their verifications, the option words which seem most likely, based on the combined ratings of all the KS's, are then used to construct new partial sentence hypotheses. The global data base is then re-evaluated to find the most promising sentence hypothesis; this hypothesis then becomes the focal point for the next hypothesize-and-test cycle.

Problems with HEARSAY I

There are four major design decisions in the HSI implementation of knowledge representation and cooperation which prevent HSI from being applied to more complex speech tasks or multiprocessor environments.

The <u>first</u>, and most important, of these limiting decisions concerns the use of the hypothesize-and-test paradigm. As implemented in HSI, the paradigm is exploited only at the word level. The implication of hypothesizing and testing at only the word level is that the knowledge representation is uniform only with respect to cooperation at that level. That is, the information content of any element in the global data base is limited to a description at the word level. The addition of non-word level KS's (i.e., KS's cooperating via either sub-word levels, such as syllables or phones, or via super-word levels, such as phrases or concepts) thus becomes cumbersome because this knowledge must somehow be related to hypothesizing and testing at the word level. This approach to non-word level KS's makes it difficult to add non-word knowledge and to evaluate the contribution of this knowledge. In addition, the inability to share non-word level information among KS's causes such information to be recomputed by each KS that needs it.

<u>Secondly</u>, HSI constrains the hypothesize-and-test paradigm to operate in a lockstep control sequence. The effect of this decision is to limit parallelism, because the time required to complete a hypothesize-and-test cycle is the <u>maximum</u> time required by any single hypothesizer KS plus the <u>maximum</u> time required by any single verifier (testing) KS. Another disadvantage of this control scheme is that it increases the time it takes the system to refocus attention, because there is no provision for any communication of partial results among KS's. Thus, for example, a rejection of a particular option word by a KS will not be noticed until all the KS's have tested all the option words.

The <u>third</u> weakness in the HSI implementation concerns the structure of the global data base: there is no provision for specifying relationships among alternative sentence hypotheses. The absence of relational structures among hypotheses has the effect of increasing the overall computation time and increasing the time to refocus attention, because the information gained by working on one hypothesis cannot be shared by propagating it to other relevant hypotheses.

The <u>fourth</u> limiting design decision relates to how a global problem-solving strategy (policy) is implemented in HSI: policy decisions, such as those involving attention focusing, are centralized (in a "Recognition Overlord"), and there is no coherent structure for the policy algorithms. The effect of having no explicit system structure for implementing policy decisions makes it very awkward to add or delete new policy algorithms and difficult to analyze the effectiveness of a policy and its interaction with other policies.

OVERVIEW OF HEARSAY II

Experience with HSI (as described above) has led to several important observations about a more general, uniform, and natural structure for representing and operating on the (dynamic) state of the utterance recognition.*

1. The internal structure of hypotheses at different levels of knowledge representation may be essentially the same, except for the primitive unit of information held in an hypothesis. This structural homogeneity in the global data base allows the actions of hypothesizing and testing at these various levels to be treated in a uniform manner.

2. The different types of knowledge (and their relationships) present in speech may be naturally represented in a single, uniform data structure. This data structure is 3-dimensional: one dimension represents information levels (e.g., phrasal, lexical, phonetic), the second represents speech time, and the third dimension contains alternative (competing) hypotheses at a particular level and time. These three dimensions form a convenient addressing structure for locating hypotheses.

3. There is a conceptually simple and uniform way of dynamically relating hypotheses at one level of knowledge to alternative hypotheses at that level and to hypotheses at other knowledge levels in the structure. The resulting structure is an AND/OR graph with modifications which provide for temporal relationships and selective dependency relationships.**

System Structure

The main goal of the HSII design is to extend the concepts developed in HSI for the representation and cooperation of knowledge at the word level to all levels of knowledge needed in a speech understanding system, based on the preceding observations.

The generalization of the hypothesize-and-test paradigm to all levels of speech knowledge implies the need for a mechanism for transferring information among levels. This mechanism is already embodied in the hypothesize-and-test paradigm; that is, one can characterize two types of hypothesization a knowledge source might be called upon to perform: horizontal and vertical hypothesization. A hypothesization is horizontal when a KS uses contextual information at a given knowledge level to predict new hypotheses at the same level, (e.g., the hypothesization that the word "night" might follow the the sequence of words "day" - "and"); whereas a hypothesization is vertical when a KS uses information at one level in the data base to predict new hypotheses at a different level (e.g., the generation of a hypothesis that a [T] occurred when a segment of silence is followed by a segment of aspiration).

* The meaning of these observations will be made more clear by the further descriptions below.

** This latter feature refers to "connection matrices" and is described below in more detail.

The HSII implementation of the hypothesize-and-test paradigm has also resulted in a generalization of the lockstep control scheme for KS sequencing employed by HSI. HSII relaxes the constraints on the hypothesize-and-test paradigm and allows the knowledge-source processes to run in an asynchronous, data-directed manner. A knowledge source is instantiated as a **knowledge-source process** whenever the data base exhibits characteristics which satisfy a "precondition" of the knowledge source. A **precondition** of a KS is a description of some partial state of the data base which defines when and where the KS can contribute its knowledge by modifying the data base. Such a modification might be adding new hypotheses proposed by the KS (at the information level appropriate for that KS) or verifying (criticizing) hypotheses which already exist. The modifications made by any given knowledge-source process are expected to trigger further knowledge sources by creating new conditions in the data base to which those knowledge sources respond. The structure of a hypothesis has been so designed as to allow the preconditions of most KS's to be sensitive to a single, simple change in some hypothesis (such as the changing of a rating or the creation of a structural link). Through this data-directed interpretation of the hypothesize-and-test paradigm, HSII knowledge sources exhibit a high degree of asynchrony and potential parallelism. A side-effect of this more general control scheme for HSII is that the strategy need not be centralized and implemented as a monolithic overlord, but rather can be implemented as **policy modules** which operate in precisely the same manner as KS's.

The 3-dimensional data base, augmented by the AND/OR structural relationships specified over that data base, permits information generated by one knowledge source to be: 1) retained for use by other KS's, and 2) quickly propagated to other relevant parts of the data base. This retention and propagation provide two important features for solving a complex problem in which errors are highly likely. First, quick refocusing can occur when a particular path no longer appears promising. Second, "selective" backtracking may be used; i.e., when a KS finds that it has made an incorrect decision, it does not have to eliminate all information generated since that decision, but rather only that subset which depends on the incorrect decision. In this way, information generated by one knowledge source is retained and is usable by itself and other KS's in other relevant contexts.

Summarizing, HSII is based on the views: 1) that the state of the recognition can be represented in a uniform, multilevel data base, and 2) that speech knowledge can be characterized in a natural manner by describing many small knowledge sources. These knowledge sources react to certain states of the data base (via their preconditions) and, once instantiated as knowledge-source processes, provide their own changes to the data base which contribute to the progress of the recognition. The hypothesize-and-test paradigm, when stated in sufficiently non-restrictive (parallel) terms, serves to describe the general interactions among these knowledge sources. In particular, changes made by one or more knowledge-source processes may trigger other knowledge sources to react to these changes by validating (testing) them or hypothesizing further changes. The intent of HSII is to provide a framework within which to explore various configurations of information levels, knowledge sources, and global strategies.*

From a more general point of view, the goal of HSII is to provide a multiprocess-oriented software architecture to serve as a basis for systems of cooperating (but independent and asynchronous) data-directed knowledge-source processes. The purpose of such a structure is to achieve effective parallel search over a general artificial intelligence problem-solving graph, employing the hypothesize-and-test paradigm to generate the search graph and using a uniform, interconnected, multilevel global data base as the primary means of interprocess communication.

HEARSAY II SYSTEM DESIGN AND IMPLEMENTATION

One can derive from the description of the desired HSII recognition process given above several basic components of the required system structure. First, a sufficiently general structured global data base is needed, through which the knowledge sources may communicate by inserting hypotheses and by inspecting and modifying the hypotheses placed there by other knowledge sources. Second, some means for describing the various knowledge sources and their internal processing capabilities is required. Third, in order to have knowledge sources activated in a data-directed manner, a method is required by which a set of preconditions may be specified and associated with each knowledge source. Fourth, in order to detect the satisfaction of these preconditions and in order to allow knowledge sources to locate parts of the data base in which they are interested, two mechanisms are needed: 1) a monitoring mechanism to record where in the data base changes have occurred and the nature of those changes, and 2) an associative retrieval mechanism for accessing parts of the data base which conform to particular patterns which are specified as matching-prototypes.

Elements of the System Structure

The following sections outline the HSII implementation of the various basic system components.

Global Data Base. The design of HSII is centered around a global data base (blackboard) which is accessible to all knowledge-source processes. The global data base is structured as a uniform, multilevel, interconnected data structure.

Each level in the data base contains a (potentially complete) representation of the utterance; the levels are differentiated by the units that make up the representation, e.g., phrases, words, phonemes. The system structure of HSII does not pre-specify what the levels in the global data structure are to be. A particular configuration, called HSII-CO (Configuration Zero), is being implemented as the first test of the HSII structure. Figure 1 shows a schematic of the levels of HSII-CO. A more detailed description and justification for this particular configuration can be found in Shockey and Erman (1974). This configuration will be used as the basis for examples to illustrate various aspects of the HSII system.

Parametric Level - The parametric level holds the most basic representation of the utterance that the system has; it is the only direct input to the machine about the acoustic signal.

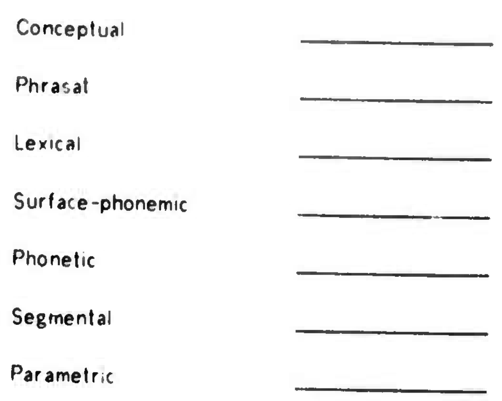


Figure 1. The Levels in HSII-CO.

Several different sets of parameters are being used in HSII-CO interchangeably: 1/3-octave filter-band energies measured every 10 msec., LPC-derived vocal-tract parameters (Knudsen, 1974), and wide-band energies and zero-crossing counts.

Segmental Level - This level represents the utterance as labeled acoustic segments. Although the set of labels may be phonetic-like, the level is not intended to be phonetic -- the segmentation and labeling reflect acoustic manifestation and do not, for example, attempt to compensate for the context of the segments or attempt to combine acoustically dissimilar segments into (phonetic) units.

As with all levels, any particular portion of the utterance may be represented by more than one competing hypothesis (i.e., multiple segmentations and labelings may co-exist).

Phonetic Level - At this level, the utterance is represented by a phonetic description. This is a broad phonetic description in that the size (duration) of the units is on the order of the "size" of phonemes; it is a fine phonetic description to the extent that each element is labeled with a fairly detailed allophonic classification (e.g., "stressed, nasalized [I]").

Surface-Phonemic Level - This level, named by seemingly contradicting terms, represents the utterance by phoneme-like units, with the addition of modifiers such as stress and boundary (word, morpheme, syllable) markings.

Syllabic Level - The unit of representation here is the syllable.

Lexical Level - The unit of information at this level is the word. (Note again that at any level competing representations can be accommodated.)

Phrasal Level - Syntactic elements appear at this level. In fact, since a level may contain arbitrarily many "sub-levels" of elements structured as a modified AND/OR graph, traditional kinds of syntactic trees can be directly represented here.

Conceptual Level - The units at this level are "concepts." As with the phrasal level, it may be appropriate to use the graph structure of the data base to indicate relationships among different concepts.

The basic unit in the data structure is a node; a node represents the hypothesis that a particular element exists in the utterance. For example, an hypothesis at the phonetic level may be labeled as 'T'. Besides containing the hypothesis element name,

* It is interesting to note that this generalized form of hypothesize-and-test leads to a system organization with some characteristics similar to QA4 (Rulifson, et al., 1973) and PLANNER (Hewitt, 1972). In particular, there are strong similarities in the data-directed sequencing of processes.

a node holds several other kinds of information, including: 1) a correlation with a particular **time period** in the speech utterance,* 2) **scheduling parameters** (validity ratings, attention focus factors, measures of computing effort expended, etc.), and 3) **connection information** which relates the node to other nodes in the data base.

Structural relationships between nodes (hypotheses) are represented through the use of **links**; links provide a means for specifying contextual abstractions about the relationships of hypotheses. A link is an element in the data structure which associates two nodes as an ordered pair; one of the nodes is termed the **upper hypothesis**, and the other is called the **lower hypothesis**. The lower hypothesis is said to **support** the upper hypothesis; the upper hypothesis is called a **use** of the lower one. There are several types of links; in general, if a node serves as the upper hypothesis for more than one link, all of those links must be of the same type. Thus, one can talk of the "type of the hypothesis," which is the same as the type of all of its lower links.

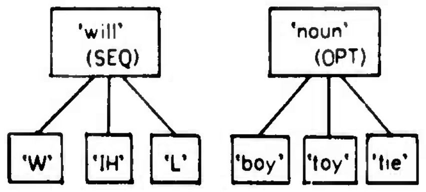


Figure 2. Examples of SEQUENCE and OPTION Relationships.

The two most important structural relationship types are SEQUENCE and OPTION:

A **SEQUENCE** node is an hypothesis that is supported by a (timewise) sequential set of hypotheses at a lower level (or sublevel -- see below). For example, Figure 2a shows an hypothesis of 'will' at the lexical level supported by the (time-)ordered contiguous sequence of 'W', 'IH', and 'L' at the surface-phonemic level. The interpretation of a SEQUENCE node is that all of the lower hypotheses must be valid in order to support the upper hypothesis.

An **OPTION** node is an hypothesis that has alternative supports from two or more hypotheses, each of which covers essentially the same time period. For example, Figure 2b shows the hypothesis of 'noun' at the phrasal level as being supported by any of 'boy', 'toy', or 'tie', all of which are competing word hypotheses covering (approximately) the same time area.**

Figure 3 is an example of a larger fragment of the global data structure. The level of an hypothesis is indicated by its vertical position; the names of the levels are given on the left.

* This time period is specified with an explicit estimation of fuzziness, even to the extent of spanning the entire utterance.

** In addition to SEQUENCE and OPTION, there are two kinds of structural relationships which are generalizations of them: An **AND** node is similar to a SEQUENCE node except that there is no implication of any time sequentiality among the supports -- they may overlap or be disjoint. An **OR** node is similar to an OPTION node in that the supports represent alternatives, but (as with the AND node) there is no time requirement.

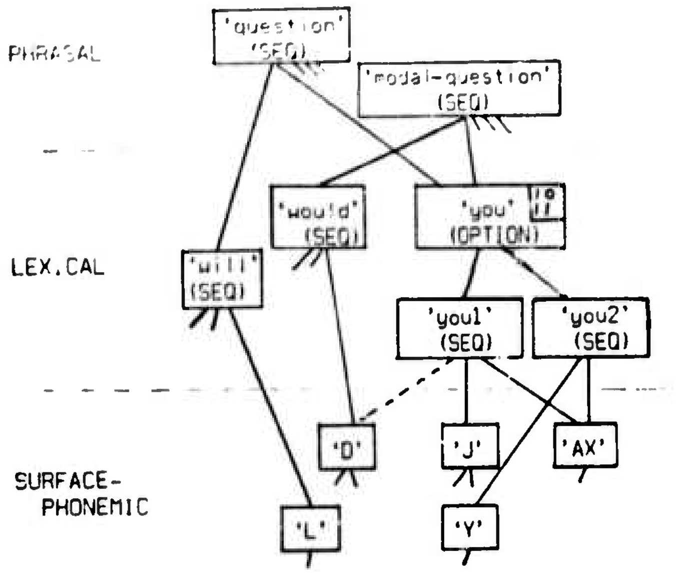


Figure 3. An Example of the Data Structure.

Time location is approximately indicated by horizontal placement, but duration is only very roughly indicated (e.g., the boxes surrounding the two hypotheses at the phrasal level should be much wider). Alternatives are indicated by proximity; for example, 'will' and 'would' are word hypotheses covering the same time span. Likewise, 'question' and 'modal-question', 'you1' and 'you2', and 'J' and 'Y' all represent pairs of alternatives.

This example illustrates several features of the data structure:

The hypothesis 'you', at the lexical level, has two alternative phonemic "spellings" indicated; the hypotheses labeled 'you1' and 'you2' are nodes created, also at the lexical level, to hold those alternatives. In general, such **sub-levels** may be created arbitrarily.

The link between 'you1' and 'D' is a special kind of SEQUENCE link (indicated here by a dashed line) called a **CONTEXT** link; a CONTEXT link indicates that the lower hypothesis supports the upper one and is contiguous to its brother links, but it is not "part of" the upper hypothesis in the sense that it is not within the time interval of the upper hypothesis -- rather, it supplies a context for its brother(s). In this case, one may "read" the structure as stating "'you1' is composed of 'J' followed by 'AX' (schwa) in the context of the preceding 'D'." (This reflects the phonological rule that "would you" is often spoken as "would-ja.") Thus, a CONTEXT link allows important contextual relationships to be represented without violating the implicit time assumptions about SEQUENCE nodes.

The phonemic spelling of the word "you" held by 'you1' includes a contextual constraint (as just described); the 'you2' option does not have this constraint. However, 'you1' and 'you2' are such similar hypotheses that there is strong reason for wanting to retain them as alternative options under 'you' (as indicated in Figure 3), rather than representing them unconnectedly. In general, the problem is that the use of an hypothesis implies certain contextual assumptions about its environment, while the support of an hypothesis may itself be predicated on a particular set of contextual assumptions. A mechanism, called a **connection matrix**, exists in the data structure to represent this kind of relationship by specifying, for an OPTION hypothesis, which of its alternative supports are applicable ("connected to") which of its uses. In this example, the connection matrix of 'you' (symbolized in Figure 3 by the 2-dimensional binary matrix in the node) specifies that support 'you1' is relevant to use 'question' (but not to 'modal-question') and that support 'you2' is relevant to both uses. The use of a connection matrix allows the efforts of preceding KS decisions to be accumulated for future use and modification without necessitating contextual duplication of parts of the data base. Thus, much of the duplication of effort due to the backtracking mode of HSI is avoided in HSII.

Besides showing structural relationships (i.e., that one unit is composed of several other units), a link is a statement about the degree to which one hypothesis implies (i.e., gives evidence for the existence of) another hypothesis. The strength of the implication is held as information on the link. The sense of the implication may be negative; that is, a link may indicate that one hypothesis is evidence for the invalidity of another. Finally, this statement of implication is bi-directional; the existence of the upper hypothesis may give credence to the existence of the lower hypothesis and vice versa.

The nature of the implications represented by the links provides a uniform basis for propagating changes made in one part of the data structure to other relevant parts without necessarily requiring the intervention of particular knowledge sources at each step. Considering the example of Figure 3, assume that the validity of the hypothesis labeled 'J' is modified by some KS (presumably operating at the phonetic level) and becomes very low. One possible scenario for rippling this change through the data base is:

First, the estimated validity of 'you1' is reduced, because 'J' is a lower hypothesis of 'you1.'

This, in turn, may cause the rating of 'you' to be reduced.

The connection matrix at 'you' specifies that 'you1' is not relevant to 'modal-question,' so the latter hypothesis is not affected by the change in rating of the former. Notice that the existence of the connection matrix allows this decision to be made locally in the data structure, without having to search back down to the 'D' and 'J.'

'Question,' however, is supported by 'you1' (through the connection matrix at 'you'), so its rating is affected.

Further propagations can continue to occur, perhaps down the other SEQUENCE links under 'question' and 'you1.'

Knowledge Source Specification. A knowledge source is specified in three parts: 1) the conditions under which it is to be activated (in terms of the data base conditions in which it is interested, as described in the section on "preconditions" below), 2) the kinds of changes it makes to the global data base, and 3) a procedural statement (program) of the algorithm which accomplishes those changes. A knowledge source is thus defined as possessing some processing capability which is able to solve some subproblem, given appropriate circumstances for its activation.

The decomposition of the overall recognition task into various knowledge sources is regarded as being a natural decomposition. That is, the units of the decomposition represent those pieces of knowledge which can be distinguished and recognized as being somehow naturally independent.* Given a sufficient set of such knowledge sources (that is, a set that provides enough overall connectivity among the various levels of the data base that a final recognition can be attained), the collection is called the "overall recognition system." Such a scheme of "inverse decomposition" (or, composition) seems very natural for many problem-solving tasks, and it fits well into the hypothesize-and-test approach to problem-solving. As long as a sufficient "covering set" for the data base connections is maintained, one can freely add new knowledge sources, or replace or delete old ones. Each knowledge source is in some sense self-contained, but each is expected to cooperate with the other knowledge sources that happen to be present in the system at that time.

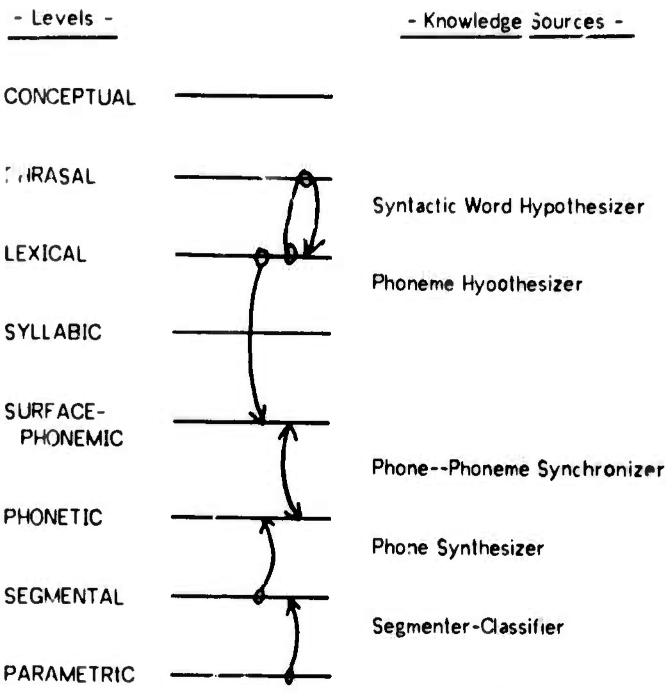


Figure 4. The First Knowledge Sources for HSII-CO.

As examples of knowledge sources, Figure 4 shows the first set of processes implemented for HSII-CO. The levels are indicated as horizontal lines in the figure and are labeled at the left. The knowledge sources are indicated by arcs connecting levels; the starting point(s) of an arc indicates the level(s) of major "input" for the KS, and the end point indicates the "output" level where the knowledge source's major actions occur. In general, the action of most of these particular knowledge sources is to create

* The approach taken in knowledge source decomposition is not an attempt to characterize somehow the overall recognition process and then apply some sort of traffic flow analysis to its internal workings in order to decompose the total process into minimally interacting knowledge sources. Rather, knowledge sources are defined by starting with some intuitive notion about the various pieces of knowledge which could be incorporated in a useful way to help achieve a solution.

links between hypotheses on its input level(s) and: 1) existing hypotheses on its output level, if appropriate ones are already there, or 2) hypotheses that it creates on its output level.

The Segmenter-Classifier knowledge source uses the description of the speech signal to produce a labeled acoustic segmentation. (See Goldberg, et al., (1974) for a description of the algorithm being used.) For any portion of the utterance, several possible alternative segmentations and labels may be produced.

The Phone Synthesizer uses labeled acoustic segments to generate elements at the phonetic level. This procedure is sometimes a fairly direct renaming of an hypothesis at the segmental level, perhaps using the context of adjacent segments. In other cases, phone synthesis requires the combining of several segments (e.g., the generation of [t] from a segment of silence followed by a segment of aspiration) or the insertion of phones not indicated directly by the segmentation (e.g., hypothesizing the existence of an [t] if a vowel seems velarized and there is no [t] in the neighborhood). This KS is triggered whenever a new hypothesis is created at the segmental level.

The Syntactic Word Hypothesizer uses knowledge at the phrasal level to predict possible new words at the lexical level which are adjacent (left or right) to words previously generated at the lexical level. (Rich (1974) contains a description of the probabilistic syntax method being used here.) This knowledge source is activated at the beginning of an utterance recognition attempt and, subsequently, whenever a new word is created at the lexical level.

The Phoneme Hypothesizer knowledge source is activated whenever a word hypothesis is created (at the lexical level) which is not yet supported by hypotheses at the surface-phonemic level. Its action is to create one or more sequences at the surface-phonemic level which represent alternative pronounciations of the word. (These pronounciations are currently pre-specified as entries in a dictionary.)

The Phone-Phoneme Synchronizer is triggered whenever an hypothesis is created at either the phonetic or the surface-phonemic level. This KS attempts to link up the new hypothesis with hypotheses at the other level. This linking may be many-to-one in either direction.

Figure 5 shows the initial HSII-CO knowledge sources of Figure 4 augmented with four additional ones which are being implemented or are planned.

The Semantic Word Hypothesizer uses semantic and pragmatic information about the task (news retrieval, in this case) to predict words at the lexical level.

The Word Candidate Generator uses phonetic information (primarily just at stressed locations and other areas of high phonetic reliability) to generate word hypotheses. This will be accomplished in a two-stage process, with a stop at the syllabic level, from which lexical retrieval is more effective.

The Phonological Rule Applier rewrites sequences at the surface-phonemic level. This KS is used: 1) to augment the dictionary lookup of the Phoneme Hypothesizer, and 2) to handle word boundary conditions that can be predicted by rule.

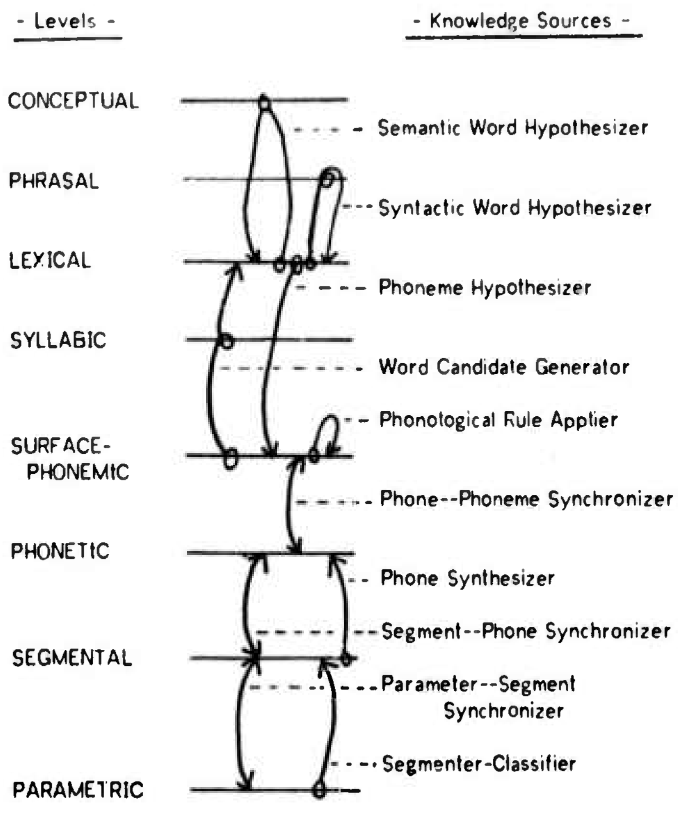


Figure 5. Augmented Knowledge Source Set for HSII-CO.

The primary duties of the Segment-Phone Synchronizer and the Parameter-Segment Synchronizer are similar: to recover from mistakes made by the (bottom-up) actions of the Phone Synthesizer and Segmenter-Classifier, respectively, by allowing feedback from the higher to the lower level.

The introduction of these knowledge sources indicates the modularity and extendability of the system in terms of both knowledge sources and levels. In particular, notice that even though the purpose of some KS is stated as "correcting errors produced by other knowledge sources," each KS is independent of the others. Yet additional knowledge sources will be added to the configuration as the need for them is seen and as ideas for their implementation are developed.

Matching-Prototypes and Associative Retrieval. The asynchronous processing activity in HSII is primarily data-directed; this implies the need for some mechanism whereby one can retrieve parts of the global data base in an associative manner. HSII provides primitives for associatively searching the data base for hypotheses satisfying specified conditions (e.g., finding all hypotheses at the phonetic level which contain a vowel within a certain time range). The search condition is specified by a **matching-prototype**, which is a partial specification of the components of a hypothesis. This partial specification permits a component to be characterized by: 1) a set of desired values, or 2) a don't-care condition, or 3) values of components of a hypothesis previously derived by matching another prototype. A matching-prototype can be compared against a set of hypotheses. Those hypotheses whose component values match those specified by the matching-prototype are returned as the result of the search. Associative retrieval of structural relationships among hypotheses is also provided by several primitives. More compl retrievals can be accomplished by combining the retrieval primitives in appropriate ways.

Preconditions and Change Sets. Associated with every knowledge source is a specification of the data base conditions required for the instantiation of that knowledge source. Such specifications, called **preconditions**, conceptually form an AND/OR tree composed of matching-prototypes and structural relationships which, when applied to the data base in an associative manner, detect the regions of the data base in which the knowledge source is interested (if the precondition is capable of being satisfied at that time). Alternatively, one might think of the precondition specification as a procedure, involving matching-prototypes and structural relationships, which effectively evaluates a conceptual AND/OR tree. This procedure may contain arbitrarily complex decisions (based on current and past modifications to the data base) resulting in the activation of desired knowledge sources within the chosen contexts. The context corresponding to the discovered data base region which satisfies some knowledge source's precondition is used as an initial context in which to instantiate that knowledge source as a new process. If there are multiple regions in the data base that satisfy the specified conditions, the knowledge source can be separately instantiated for each context, or once with a list of all such contexts.

Whenever any KS performs a modification to the global data base, the essence of the modification is preserved in a **change set** appropriate to the change made (e.g., one change set records rating changes, while another records time range changes). Change sets serve to categorize data base modifications (events) and are thus useful in precondition evaluation since they limit the areas in the data base that need to be examined in detail. As currently implemented, precondition evaluators exist as a set of procedures which monitor changes in the data base (i.e., they monitor additions to those change sets in which they are interested). These precondition procedures are themselves data-directed in that they are applied whenever sufficient changes have been made in the global data base. In effect, the precondition procedures themselves have preconditions, albeit of a much simpler form than those possible for knowledge sources. For example, a precondition procedure may specify that it should be invoked (by a system precondition invoker) whenever changes occur to two adjacent hypotheses at the word level or whenever support is added to the phrasal level. By using the (coarse) classifications afforded by change sets, the system avoids most unnecessary data base examinations by the precondition procedures. The major point to be made is that the scheme of precondition evaluation is **event-driven**, being based on the occurrence of changes in the global data base. In particular, precondition evaluators are not involved in a form of busy waiting in which they are constantly looking for something that is not yet there.

Once invoked, a precondition procedure uses sequences of associative retrievals and structural matches on the data base in an attempt to establish a context satisfying the preconditions of one or more of "its" knowledge sources; any given precondition procedure may be responsible for instantiating several (usually related) knowledge sources. Notice that the data-directed nature of precondition evaluation and knowledge-source instantiation is linked closely to the primitive functions that are able to modify the data base, for it is only at points of modification that a precondition that was unsatisfied before may become satisfied. Hence, data base modification routines have the responsibility (although perhaps indirectly) of activating the precondition evaluation mechanism.*

Multiprocessing Considerations: Some Problems and Their Solutions

A primary goal in the design of HSII is to exploit the potential parallelism of the Hearsay system model as fully as possible. Several issues associated with the introduction of parallel knowledge-source processes into HSII will be described and their current solutions outlined.

Local Contexts. A precondition evaluator (process) is invoked based on the occurrence of certain changes which have taken place in the global data base since the last time the evaluator was invoked; these changes, together with the state of the relevant parts of the global data base at the instant at which the precondition evaluator is invoked, form a **local context** within which the evaluator operates. Conceptually, at the instant of its invocation, the precondition evaluator takes a snapshot of the global data base and saves the substance of the changes that have occurred to that moment, thereby preserving its local context.

The necessity of preserving this local context exists for several reasons: 1) HSII consists of asynchronous processes sharing a common global data base which contains only the most current data (that is, no history of data modification is preserved in the global data base), 2) since the precondition evaluators are also executed asynchronously, each evaluator is interested only in changes in the data base which have occurred since the last time that particular evaluator was executed (that is, the relevant set of changes for a particular precondition evaluator is time-dependent on the last execution of that evaluator), and 3) further modifications to the global data base which are of interest to a given knowledge-source process may occur between the time of that knowledge-source process's instantiation and the time of its completion (in particular, such modifications and their relationship to data base values which existed at the time of the instantiation of the knowledge-source process may influence the computation of that knowledge-source process). Hence, the problem of creation of local contexts exists, as do the associated problems of signalling a knowledge-source process when its local context is no longer valid and of updating these contexts as further changes occur in the global data base.

Consider the following time sequence of events:

* One might think of HSII as a production system (Newell, 1973) which is executed asynchronously. The preconditions correspond to the left-hand sides (conditions) of productions, and the knowledge sources correspond to the right-hand sides (actions) of the productions. Conceptually, these left-hand sides are evaluated continuously. When a precondition is satisfied, an instantiation of the corresponding right-hand side of its production is created; this instantiation is executed at some arbitrary subsequent time (perhaps subject to instantiation scheduling constraints).

T1: start precondition evaluator PRE
(triggered by data base changes)
T2: PRE instantiates a knowledge-source process KS
T3: end PRE

T4: start KS
T5: after KS revalidation of precondition
<computation>
T6: KS modifies global data base
<computation>
T7: KS modifies global data base again
T8: end KS

PRE is activated to respond to changes occurring in the global data base. PRE should execute in the context of changes existing at time T1 (since that context contains the changes which caused PRE to be activated). KS is instantiated (readied for running) at T2 due to further conditions PRE discovered about the change context of T1. Hence, PRE should pass the context of T1 as the initial environment in which to run KS.

By time T4, when KS actually starts to execute, other changes could have occurred in the global data base due to the actions of other knowledge-source processes. So KS should examine these new updating changes (those occurring between T1 and T4) and revalidate its precondition, if necessary. After revalidation, KS assumes the updated context of T5, and it proceeds to base its computation on the context of changes as of T5.

When KS wishes to perform an actual update of elements of the global data base at T6, it must examine the changes to the global data base that occurred between T5 and T6 to see if any other knowledge-source processes may have violated KS's preconditions, thereby invalidating its computations. Having performed this revalidation and any data base updating, KS should update its context to reflect changes up to T6 for use in its further computation. At T7, KS must look for further possible invalidations to its most recent computations, due to possible changes in the global data base by other knowledge-source processes during the time period T6 to T7. When KS (which is an instantiation of some knowledge source) completes its actions at T8, its local context may be deleted.

Changes occurring between instantiations of a knowledge source are accumulated in the local contexts of the precondition evaluators and may become part of the local context of a future instantiation of a knowledge source. Thus, the precondition evaluators are always collecting data base changes (since these evaluators are permanently instantiated), while individual knowledge source instantiations accumulate data base changes only during their transient existence.

In practice, to create local contexts one need only save the contents of changes which occur in the global data base (thereby allowing the global data base to contain only the very latest values). In particular, no massive copy operations involving the global data base are required. Thus, for each data base event caused by a modification primitive, the associated change* is distributed (copied) into the local contexts (which can now be characterized as local **change sets**,* referring to the previous discussion on change sets) of all knowledge-source processes and precondition evaluators who care. Notice that not every knowledge-source process and precondition evaluator cares about every change that takes place. For example, a fricative detector will not care about a data base change associated with the grouping of several words to form a syntactic phrase, but it is interested in the hypothesization of a word whose phonemic spelling contains a fricative.

In order for a knowledge-source (or precondition evaluator) process to receive changes which occur to particular fields of particular nodes which are in its local context, those fields need to be **tagged** with that knowledge-source instantiation's unique name. Then, whenever a modification is done to the global data base, a message signalling the change is sent to all who have tagged the field being changed. In addition, the contents of the change are also distributed to the local contexts of those knowledge sources receiving the message. This data field tagging may be requested by either a precondition evaluator which is about to instantiate a knowledge source based on the values of particular fields (which represent the context satisfying the precondition), or by a knowledge-source process, once instantiated, which may request additional fields to be tagged.

For example, in its search through the global data base for conditions satisfying the preconditions of some knowledge source, a precondition evaluator may accumulate references to the data fields which it has examined; and when the entire precondition has been found to be satisfied, the precondition evaluator tags those fields (for which it has accumulated references) with the name of the knowledge-source process it is about to instantiate. Similarly, having been invoked, a knowledge-source process might wish to do certain computations, but only if certain data fields are not altered; the knowledge-source process itself can tag these fields and thereby be informed of subsequent tampering with the tagged fields. The union of these tagged fields forms a **critical set** (specifying the fields of the local context for the knowledge-source process) which is locked (see below) every time the knowledge-source process wishes to modify the global data base. Thus, after having locked the critical set and prior to performing any update operations, the knowledge-source process can check to see whether any other knowledge-source process has made any changes which might invalidate the current knowledge source's assumed context (and hence, perhaps, invalidate its proposed update).** For example, if a knowledge-source process is verifying a hypothesis in the data base because its rating exceeds 50 (i.e., the rating value represents part of the local context for the knowledge-source process), then before

* The information which defines the change consists of the locus of the change (i.e., a node name and a field name), the old value of the field, the revised value of the field, the name of the changer (the unique knowledge-source instantiation name), and the system time of the change.

* An alternative to replicating the change information for the various knowledge-source processes is to maintain a single central copy of those data, passing only references to the centralized items to the various local contexts. The individual change items may then be deleted from their central change sets whenever there are no further outstanding references to that change.

** The characterization of local contexts according to specific data fields (which is made possible in part by the choice of levels in the global data base) helps to minimize the overhead associated with context maintenance. Also, the smaller the context, the more flexible the scheduling strategy may be (since it needs to be less concerned with the time required for a context swap on a processor).

performing any modifications on the data base which depend on that assumption, the knowledge-source process should check to be sure that no other knowledge-source process has invalidated the rating assumption in the meantime.

Data Base Deadlock Prevention. Any knowledge-source process may request exclusive access to some collection of fields at any time. Thus, unless some care is taken in ordering the requests for such exclusive access, the possibility of getting into a deadlock situation exists (where, for example, one knowledge-source process is waiting for exclusive access to a field already held exclusively by a second knowledge-source process, and *vice versa*, resulting in neither process being able to proceed). Applying a linear ordering to the set of lockable fields and requesting exclusive access according to that ordering is a commonly-used means of deadlock avoidance in resource allocation. However, this technique works only if all the resources (fields) to be locked are known ahead of time. The ability to tag a data field allows a knowledge-source process to locate and examine in arbitrary order the set of hypotheses that will form the context for a data base modification and then, only after the entire set of desired hypotheses (and links) has been determined, to lock the desired set and perform the modification.

To eliminate the possibilities of deadlock, the actual locking operation is relegated to a system primitive, and a knowledge-source process is required to present to the locking primitive the entire set of fields to which it wants exclusive access. This set is then extended to include all fields in the critical set of the calling knowledge-source process, for the reasons relating to context revalidation given above. The system locking primitive can then request exclusive access for this union of data fields, on behalf of the knowledge-source process, according to a universal ordering scheme (such an ordering being possible since every node in the global data base essentially has a unique serial number) which assures that no deadlocks occur. Having been granted exclusive access to all desired fields at once, the knowledge-source process may first check to see whether there have been any changes to the tagged data fields. If there have been none, it can proceed to perform its modifications (which modifications are sent to the local contexts of others who care about such things). However, if there were changes, the knowledge-source process can, after examining the changed data fields, decide whether it still wants to perform the modifications. When the knowledge-source process is finished updating the data base, it releases all its locked fields by executing a system primitive unlocking operator. In particular, the system ensures that a knowledge-source process will not request two lock operations without issuing an intervening unlock operation. In this manner, any possibility of a data base deadlock is eliminated.

Goal-Directed Scheduling. The computational sequence of a typical HSII run may be characterized by considering the states of the utterance recognition at any particular data base level. For example, if one considers the efforts in producing the word level and traces the development of the "best" partial sentences, the processing that will have been done is analogous to a general tree-search, where each node of the tree represents some partially-completed sentence (with the eventual resultant sentence being one of the leaves of the tree). The problem now is to guide this tree searching so as to find the answer leaf in an optimal way (according to some measure of optimality). To achieve this end, ratings are associated with every hypothesis and link in the global data base (and thus with every partial sentence node of the analogous search tree). Using these ratings (which are effectively evaluation functions indicating the goodness of the results of the work done so far with respect to a given partial sentence), one may employ various tree-searching strategies to advance the search in some optimal manner.

To complicate matters further, however, HSII is intended to be a multiprocess-oriented system. Therefore, schemes must be devised for effectively searching a problem-solving graph using parallel processes, since one can conceive of pursuing several branches of the search graph in parallel by asynchronously instantiating various knowledge-source processes to evaluate various alternative paths. One must also take into consideration the underlying hardware architecture. The physical placement of the global data base and the knowledge-source processes will have a very definite influence on the scheduling philosophy chosen and the resultant system efficiency.

To aid in making scheduling decisions, one may associate with every node in the global data base some **attention focusing factors**, which are indicators telling how much effort has been devoted to processing in this area of the search tree and how desirable it is to devote further effort to this section of the tree. Such attention focusing factors may also be associated with various speech time regions to indicate interest in doing further processing on certain regions of the utterance, regardless of any particular information level. Furthermore, attention focusing factors are associated with every data base modification, thereby distributing attention focusing factors to the various change sets which constitute the local contexts of the processes in the system. The scheduler is one such process which might be especially interested in such focusing factors, as will be described below. The use of the various ratings and attention focusing factors allows HSII to perform **goal-directed scheduling,** which is process scheduling so as to achieve "optimal" recognition efficiency. The thrust of goal-directed scheduling is that, while there may be many processes ready to run and work on various parts of the search tree, one should first schedule those processes which can best help to achieve the goal of utterance recognition. Notice that such search-tree pruning techniques as the alpha-beta procedure (which is essentially a sequential algorithm anyway) do not apply to HSII's non-game search trees, which do not have the constraint that alternating levels of the tree represent the moves of an opponent.

Goal-directed scheduling may be viewed as having two separate functions: 1) using the ratings and attention focusing factors associated with the global data base components to schedule _knowledge-source processes_ which have been invoked (readied for execution) in response to previous events detected in the global data base, and 2) using these same attention focusing factors to detect important areas in the global data base which require further work, and invoking _precondition evaluators_ as soon as possible to instantiate new knowledge-source processes to work in those important areas. Thus, the attention focusing factors within the global data base serve to schedule both knowledge-source processes and precondition evaluators.

A knowledge-source process might be scheduled for execution because it possesses the only processing capability available to be applied to an important unexplored area of the data base. However, if there are many such processes ready to execute, the scheduler can perform a type of **means-ends analysis** in which those knowledge-source processes are scheduled which are likely to produce data base changes in which the system is currently interested (such interest being noted by high attention focusing factors in a given change set). For example, if the data base contains focusing factors which highlight activity in a time region in which there are no structural connections between two adjoining levels, the scheduler would probably give a higher priority to a knowledge-source process which will attempt (as indicated in its external specifications) to make such a connection than to a knowledge-source process which is likely merely to perform a minor refinement on the ratings in one of the levels.

Another means of effecting goal-directed scheduling relates to the attention focusing factors associated with various time regions of the utterance (such focusing factors reaching across all the information levels of the global data base). Using these time region focusing factors, one can schedule knowledge-source processes which can contribute in a particular time region, or invoke precondition evaluators to instantiate some new knowledge-source processes to work within the desired time region.

SUMMARY, CURRENT STATUS, AND PLANS

In summary, HSII is a system organization for speech understanding that permits the representation of speech knowledge in terms of a large number of diverse knowledge sources which cooperate via a generalized (in terms of both data and control) form of the hypothesize-and-test paradigm. Knowledge sources are independent and separable; they are activated in a data-directed manner and execute asynchronously, communicating information among themselves through a global data base. This global data base, which is a representation of the partial analysis of the utterance, is a three-dimensional data structure (in which the dimensions are level of representation, time, and alternatives) augmented by AND/OR structural relationships which interconnect elements of the data structure. This global data base structure permits information generated by one knowledge source to be: 1) retained for use by other knowledge sources, and 2) quickly propagated to other relevant parts of the data base. In addition to being a new representational framework for specifying speech knowlege, HSII is a system organization suitable for efficient implementation on a multiprocessor computer system. In particular, the system organization employs techniques which, while not violating the independence and modularity of knowledge sources, permits: 1) avoidance of deadlock in the data base, 2) efficient implementation of data-directed sequencing of knowledge sources, and 3) goal-directed scheduling of asynchronously executable knowledge-source processes.

A preliminary, synchronous version of HSII has been operating on CMU's PDP-10 since January, 1974. The fully asynchronous, multiprocess version of HSII is now in the final stages of being implemented, also on the PDP-10, and is expected to be running by May, 1974. This multiprocess version of HSII will also contain the capability of simulating the effect of operating HSII in a multiprocessor environment. Experience with this multiprocess version of HSII, together with simulation data on the effects of operating in a multiprocessor environment, will form the basis for a multiprocessor version of HSII on C.mmp. An initial implementation of HSII on C.mmp is expected to be completed in the first quarter of 1975.

ACKNOWLEDGMENTS

We wish to acknowledge the contributions of the following people: Allen Newell for his insights into the task of problem-solving, David Parnas and Lee Cooprider for their discussions on system design and decomposition, Gregory Gill for his untiring efforts in system implementation, and Donald McCracken for his valuable criticism of the ideas as presented in this paper.

BIBLIOGRAPHY

Bell, C. G., W. Broadley, W. Wulf, A. Newell, et al., (1971), "C.mmp: The CMU Multi-mini-processor Computer," Tech. Rep., Comp Sci. Dept., Carnegie-Mellon Univ., Pittsburgh, Pa.

Bell, C. G., R. C. Chen, S. H. Fuller, J. Grason, S. Rege, and D. P. Siewiorek (1973), "The Architecture and Application of Computer Modules: A Set of Components for Digital Systems Design," COMPCON 73, San Francisco, Ca., pp. 177-180.

Erman, L. D., R. D. Fennell, V. R. Lesser, and D. R. Reddy (1973), "System Organizations for Speech Understanding: Implications of Network and Multiprocessor Computer Architectures for AI," Proc. 3rd Inter. Joint Conf. on Artificial Intel., Stanford, Ca., pp. 194-199.

Goldberg, H. G., D. R. Reddy, and R. L. Suslick (1974), "Parameter-Independent Machine Segmentation and Labeling," Proc. IEEE Symp. Speech Recognition, Pittsburgh, Pa., pp. 106-111, (this volume).

Hewitt, C. (1972), "Description and Theoretical Analysis (Using Schemata) of Planner: A Language for Proving Theorems and Manipulating Models in a Robot," AI Memo No. 251, MIT Project MAC.

Knudsen, M. J. (1974), "Real-time Linear-predictive Coding of Speech on the SPS-41 Microprogrammed Triple-processor System," Proc. IEEE Symp. Speech Recognition, Pittsburgh, Pa., pp. 274-277, (this volume).

Kriz, S. (1974), "A 16-bit A-D-A Conversion System for High-Fidelity Audio Research," Proc. IEEE Symp. Speech Recognition, Pittsburgh, Pa., pp. 278-282, (this volume).

Newell, A., J. Barnett, J. Forgie, C. Green, D. Klatt, J. C. R. Licklider, J. Munson, R. Reddy, and W. Woods (1971), **Speech Understanding Systems: Final Report of a Study Group**, pub. by North-Holland (1973).

Newell, A. (1973), "Production Systems: Models of Control Structures," in W. C. Chase (ed.) **Visual Information Processing**, Academic Press, pp. 463-526.

Reddy, D. R., L. D. Erman, and R. B. Neely (1970), "The C-MU Speech Recognition Project," Proc. IEEE System Sciences and Cybernetics Conf., Pittsburgh, Pa.

Reddy, D. Raj, Lee D. Erman, and Richard B. Neely (1973a), "A Model and a System for Machine Recognition of Speech," **IEEE Trans. Audio and Electroacoustics**, AU-21, 3, June, 1973, pp. 229-238.

Reddy, D. R., L. D. Erman, R. D. Fennell, and R. B. Neely (1973b), "The HEARSAY Speech Understanding System: An Example of the Recognition Process," Proc. 3rd Inter. Joint Conf. on Artificial Intel., Stanford, Ca., pp. 185-193.

Reddy, R., and A. Newell (1974), "Knowledge and its Representation in a Speech Understanding System," in L. W. Gregg (ed.) **Knowledge and Cognition**, Lawrence Erlbaum Assoc., Washington, D. C., chap. 10, (in press).

Rich, E. (1974), "Inference and Use of Simple Predictive Grammars," Proc. IEEE Symp. Speech Recognition, Pittsburgh, Pa., p. 242, (this volume).

Rulifson, J. F., et al., (1973), "QA4. A Procedural Calculus for Intuitive Reasoning," Technical Note 73, Stanford Res. Inst., AI Center.

Shockey, L. and L. D. Erman (1974), "Sub-Lexical Levels in the HearsayII Speech Understanding System," Proc. IEEE Symp. Speech Recognition, Pittsburgh, Pa., pp. 208-210, (this volume).

THE DRAGON SYSTEM-- AN OVERVIEW

James K. Baker
Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa.

ABSTRACT

This paper briefly describes the major features of the DRAGON speech understanding system. DRAGON makes systematic use of a general abstract model to repesent each of the knowledge sources necessary for automatic recognition of continuous speech. The model--that of a probabilistic function of a Markov process--is very flexible and leads to features which allow DRAGON to function despite high error rates from individual knowledge sources. Repeated use of a simple abstract model produces a system which is simple in structure, but powerful in capabilities.

INTRODUCTION

To achieve reliable speech recognition it is necessary to combine information from a variety of sources([4]). In addition to the direct acoustic information, valuable sources include the vocabulary, the grammar, and the semantics of the utterance. Extracting the information from each of these sources of knowledge is a difficult task and the need to coordinate the various pieces of information makes the task even more difficult. For the DRAGON system a general theoretical model has been adapted to represent each of the important sources of knowledge. The sources of knowledge are organized into a hierarchical system such that the integrated system is again an instance of the same general model. The availability of this general theretical framework has greatly simplified the DRAGON speech understanding system.

The general model which is used throughout the DRAGON system is that of a probabilistic function of a Markov process([2]). In this model there are two sequences of random variables X(1), X(2), X(3), ... , X(T), and Y(1), Y(2), Y(3), ... , Y(T). The X's correspond to internal states which are not observed and the Y's correspond to external observations whose distributions depend on the values of the X's. For example, the X's could represent the sequence of phones in an utterance and the Y's could represent the sequence of acoustic measurements. Alternatively, the X's could be the sequence of words in an utterance and the Y's could represent the sequence of phones and modifiers as the words are actually pronounced. Changing the frame of reference again, the Y's could represent the words of the utterance and the X's could represent the underlying sequence of syntactic-semantic states.

Features of the DRAGON System

The major features of the DRAGON system are
(1) Delayed decisions
(2) Generative form of model
(3) Hierarchical system
(4) Integrated representation
(5) General theoretical framework

The various sources of knowledge are organized into a hierarchy of probabilistic functions of Markov processes. A network is constructed to provide an integrated representation of the hierarchy. Recognition of an utterance corresponds to finding an optimum path through the network. The optimum path is found by an algorithm which, in effect, explores all possible paths in parallel([1]).

Delayed Decisions

In terms of the network representation, most speech recognition algorithms search for a suboptimum path though the network. A globally optimum path would clearly be superior, but with most models it is prohibitively expense to compute. The Markov model of the DRAGON system permits such a globally optimum path to be found by an algorithm such that the number of computatons is linear in the length of the utterance.

The Markov assumption is a prescription to include "all relevant context" in formulating the state space of the process. By considering at each point in time all possible internal states, the algorithm searchs all possible paths through the network. By combining paths when and only when they come to the same state at the same time, all decisions are delayed until the full effect of all context, past and future, has been considered.

Generative Form of Model

By having an external sequence (Y) depend probabilistically on an unobserved internal sequence (X), the system allows knowledge sources to be represented in a generative form([6]). Given the sequence of syntactic-semantic states one can generate the words. Given the words one can generate the phones. Given the sequence of phones one can generate the sequence of acoustic observations. But, computationally, this hierarchy of conditional probabilities can be reversed by applying Bayes' theorem. In analyzing a specific utterance one can proceed from the known observations to the internal states which must be inferred.

Hierarchical System

The sources of knowledge are organized into a hierarchy based on the following observation: The "top" levels of a speech recognition system change state less frequently than the "bottom" levels. Thus a single syntactic-semantic state corresponds to a sequence of several words; a single word corresponds to a sequence of several phones; and a phone corresponds to a sequence of several acoustic events. The hierarchy is not absolute--for example, syntax and semantics are mixed together into a single multi-level process--but it provides a convenient means for combining the Markov processes which represent the individual sources of knowledge.

Integrated Representation

A network is constructed which represents the total hierarchy of Markov processes. The process as a whole fits the same general model as the pieces--it is a probabilistic function of a Markov process. All of the "knowledge" of the system is represented in a pair of simple data structures: the transition matrix of the network and the table of conditional probabilities connecting internal states to external observations. The main program of the system is based on the general model of a probabilistic function of a Markov process. All speech-specific knowledge is represented in the data structures, not in the program.

General Theoretical Framework

Having a general theoretical structure greatly simplifies the speech recognition system. It is both easier to implement and easier to understand. Its operations can be expressed explicitly by a simple set of mathematical equations. A powerful general system is constructed by repeated use of a flexible theoretical model.

Potential Problems and Disadvantages

Delayed decisions--searching all possible paths through the network--could lead to a combinatorial explosion in the number of computations. The Markov model completely prevents this combinatorial explosion. Alternate paths are recombined at exactly the same rate that new branches are formed. The total number of computations is linear in the length of the utterance.

The integrated representation of a hierarchical system could result in an excessively large state space. Care must be exercised as to what context must be included and what can be safely ignored. Experience indicates, however, that the network representation is a compact and powerful representation and speech recognition tasks with large vocabularies can be accomodated.

Representing all knowledge as conditional probabilities does not imply any loss of power, since the probabilities can be set to zero or to one whenever appropriate. However, it does require that estimates be computed for all the probabilities in the system. Fortunately, all these probabilities are easily estimated from the frequency of occurrence of corresponding events in a set of training utterances.

General Model

Let the sequence X(1), X(2), X(3), ... , X(T) be the sequence of states of a Markov process ([3]) with transition matrix $A = (a_{ij})$. Let Y(1), Y(2), Y(3), ... , Y(T) be a sequence of random variables such that, for all t, PROB(Y(t)=k | X(t-1)=i, X(t)=j) = b_{ijk}. Use a bracket and colon notation to abbreviate sequences. Thus X[1:T] = X(1), X(2), X(3), ... , X(T) and Y[1:T] = Y(1), Y(2), Y(3), ... , Y(T). The assumptions of the model are that

$$\begin{aligned} &\text{PROB}(Y(t)=y(t) \mid X[1:t]=x[1:t], Y[1:t-1]=y[1:t-1]) \quad (1)\\ &\quad = \text{PROB}(Y(t)=y(t) \mid X(t-1)=x(t-1), X(t)=x(t))\\ &\quad = b_{x(t-1)x(t)y(t)} \end{aligned}$$

and

$$\begin{aligned} &\text{PROB}(X(t)=x(t) \mid X[1:t-1]=x[1:t-1]) \quad (2)\\ &\quad = \text{PROB}(X(t)=x(t) \mid X(t-1)=x(t-1))\\ &\quad = a_{x(t-1)x(t)} \end{aligned}$$

Under these assumptions,

$$\begin{aligned} &\text{PROB}(X[1:T]=x[1:T], Y[1:T]=y[1:T]) \quad (3)\\ &\quad = \Pi_{t=1}^{T} a_{x(t-1)x(t)} b_{x(t-1)x(t)y(t)} \end{aligned}$$

where a special extra state x(0) is introduced and $a_{x(0)i}$ and $b_{x(0)ik}$ are defined appropriately.

It is convenient to introduce a special notation for the total probability of all partial sequecnes resulting in a particular state at a particular time. Let

$$\begin{aligned} &\alpha(s,j) = \text{PROB}(X(s)=j, Y[1:s]=y[1:s]) \quad (4)\\ &\quad = \Sigma_{x[1:s-1]} \Pi_{t=1}^{s} a_{x(t-1)x(t)} b_{x(t-1)x(t)y(t)} \end{aligned}$$

where the sum is over all possible sequences x[1:s-1] and x(s)=j. The values of α for a given s can easily be computed from the values for s-1. In fact,

$$\alpha(s,j) = \Sigma_i \alpha(s-1,i) a_{ij} b_{ijy(s)} \quad (5)$$

Conditional probabilities based on the known sequence y[1:T] can be computed from the function α and a similar function computed backwards in time from the end of the sequence. For example,

$$\begin{aligned} &\text{PROB}(X(T)=j \mid Y[1:T]=y[1:T]) \quad (6)\\ &= \text{PROB}(X(T)=j, Y[1:T]=y[1:T]) / \text{PROB}(Y[1:T]=y[1:T])\\ &= \alpha(T,j) / \Sigma_i \alpha(T,i). \end{aligned}$$

Each of the sources of knowledge needed for speech recognition can be represented with this general Markov framework.

Representation of Knowledge Sources

Representing Acoustic-Phonetic Knowledge

There are several choices in how to represent acoustic-phonetic knowledge. A decision must be made whether acoustic observations should be preprocessed by specialized procedures or whether the stochastic model should deal directly with the acoustic parameters. To simplify the exposition, consider just the case in which specialized preprocessing is done.

Assume that at each time t ($1 \le t \le T$), an acoustic observation is made. Each such observation consists of a vector of values of a set of acoustic parameters, which in the stochastic model is represented by a vector-valued random variable Y(t). There is a sequence of phones P[1:J] which is produced during the time interval $1 \le t \le T$. Assume that the phones occupy disjoint segments of time that is, assume there is a sequence $s_0 < s_1 < s_2 < s_3 < \dots < s_J$ such that P(j) lasts from observation $Y(s_{j-1})$ through observation $Y(s_j-1)$. (Set $s_0 = 1$, $s_J = T$.).

Let p[1:J] be the actual sequence of phones in an utterance and let y[1:T] be the actual observed sequence of acoustic parameters. For convenience, also introduce a special initialization phone p(0) which is assigned a special value to allow the initial probabilities to have the same form as the transition probabilities later in the sequence. Since the actual times $s_1, s_2, s_3, \dots, s_{J-1}$ are not known, it is necessary to associate each arbitrary segment of time with some phone. For any pair of times t_1 and t_2 let $S(t_1,t_2)$ be that value of j for which the expression $(\text{Min}(s_j,t_2)-\text{Max}(s_{j-1},t_1))$ is maximized. If $t_2 \le 1$ then set $S(t_1,t_2) = 0$.

The acoustic preprocessor tries to estimate a phonetic transcription from the acoustics alone. By looking for discontinuities or rapid changes in the acoustic parameters, the preprocessor divides the sequence Y[1:T] up into K phone-like segments $Y[1:t_1-1]$, $Y[t_1:t_2-1]$, $Y[t_2:t_3-1]$, ... , $Y[t_{K-1}:t_K]$. Then an attempt is made to classify each segment $Y[t_{k-1}:t_k-1]$ using some form of pattern recognition procedure. Let $t_0 < t_1 < t_2 < \dots < t_K$ be the segment boundary times as decided by the preprocessor and introduce the random variable D(t) which is 1 if there exists a k such that $t_k = t$ and is 0 otherwise. Let F(k) be the label assigned by the preprocessor to the segment $Y[t_{k-1}:t_k-1]$. (For completeness, set $t_k=t_0=1$ for k<0, and $t_k=t_K=T$ for k>K.)

For some pattern matching procedures it is possible to directly estimate conditional probabilities. When using such a procedure, let

$$B[p,k] = \mathrm{PROB}(Y[t_{k-1}:t_k-1]=y[t_{k-1}:t_k-1] \mid P(S(t_{k-1},t_k))=p). \quad (7)$$

The pattern matching procedure might yield only the label F(k) representing a best guess as to the underlying phone. In such a case it is necessary to estimate the conditional probabilities from statistics of performance by the pattern matcher on training data. Let f[1:K] represent the actual sequence of labels generated by the pattern recognizer for the utterance being considered. Then set

$$B[p,k] = \mathrm{PROB}(F(k)=f(k) \mid P(S(t_{k-1},t_k))=p), \quad (8)$$

where the conditional probability is estimated by the frequency of such events in a set of training utterances.

In addition to estimating the probability of substituitons or confusions, it is necessary to estimate the probability of the preprocessor producing either too many or too few segments. The probability of such events may be estimated from their frequency of occurence in a set of training utterances. Let

$$E[p_1,p_2,n] = \quad (9)$$

$$\mathrm{PROB}(\, D(t_{k-2})=D(t_{k-1})=D(t_k)=1,\ D[t_{k-2}+1:t_{k-1}-1]=0,\ D[t_{k-1}+1:t_k-1]=0 \mid$$

$$P(S(t_{k-2},t_{k-1}))=p_1,\ P(S(t_{k-1},t_k))=p_2,\ S(t_{k-1},t_k)=S(t_{k-2},t_{k-1})+n\ ).$$

If the acoustic preprocessor is reliable, then $E[p_1,p_2,n]$ should be small except to $n=1$ and should be negligible for $n>2$. In the DRAGON system, it has arbitrarily been assumed that $E[p_1,p_2,n]=0$ for $n>4$. Note that $E[p_1,p_2,0]$ is undefined and meaningless unless $p_1=p_2$.

We can now estimate the conditional probability of the sequence Y[1:T] given the sequence P[1:J].

$$\mathrm{PROB}(Y[1:T]=y[1:T] \mid P[0:J]=p[0:J]) \quad (10)$$

$$= \Sigma_{n[1:K]} \Pi_{k=1}^{K} B[p(z(k)),k]E[p(z(k-1)),p(z(k)),n(k)],$$

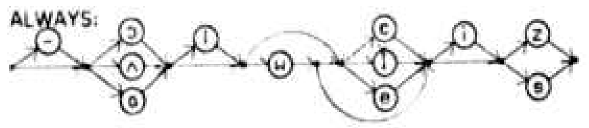
where $z(k) = \Sigma_{i=1}^{k} n(i)$ and the sum is taken over all sequences n[1:K] such that $z(K)=J$. (By convention $z(0)=0$).

In order to apply the theory of a probabilistic function of a Markov process, it is necessary to specify the transition probabilities for the phone sequence P[1:J]. It is the task of the other sources of knowledge to specify these probabilities. Phonological rules may be represented either directly or indirectly in the estimates of $E[p_1,p_2,n]$ and B[p,k], but all higher levels of the hierarchy deal only with the sequence P[1:J] and are insolated from the acoustics Y[1:T] or the labels F[1:K].

Representation of Lexical Knowledge

This section discusses the computation of the conditional probability PROB(P[1:J]=p[1:J] | W[1:I]=w[1:I]) where W[1:I] is the sequence of words in the utterance and P[1:J] is the sequence of phones. Knowledge of the sequecne of words in an utterance is such a strong determiner of the sequence of phones that it is unusual to formulate the connection as a stocastic process. Nevertheless, the stochastic formulation can represent the same rules as other formulations and in a compact and computationally convenient form.

Let's first consider how alternate pronunciations of a particular word can be represented by a probability network. As an example, take the word "always" as used in the ARCS (Automatic Recognition of Continuous Speech, IBM-Rockwell) system([9],[5]). There are 432 pronunciations which are allowed. The ARCS system can have such a complete list of phonetic variants because it uses a network representation of the alternatives and constraints. Some soeech understanding systems use an explicit list of alternate pronunciations, either generated automatically from a phonemic dictionary or preselected by hand. But an easy way to represent an exhaustive list of alternate pronunciations is by a network. The network representation for "always" is



where the dots (.) are dummy nodes introduced so that the network can be shown in two dimensions. We have represented the phones as nodes rather than as arcs (which would be even more compact) because such a representation fits more easily into the integrated system. The node-based representation permits explicit representation of sequential constraints (such as the restriction that if /w/ is omitted, then the following vowel cannot also be omitted).

The network representing alternate pronunciations of a given word can either be derived by hand and stored in a dictionary of word networks, or can be derived by automatic procedures. The automatic procedures take a canonical pronunciation and apply phonological rules to produce a network representing all likely pronunciations of the word. Even if alternate pronunciations of words are not derived by rule, the phonological rules are still important because many of them can apply across word boundaries.

The process of applying phonological rules is one way in which the DRAGON system deviates from the conceptual hierarchy. The syntax and semantics of a particular task is represented by a network in which each node corresponds to a word. Using either a dictionary of canonical pronunciations or a word-network dictionary, a small network is substituted for each word-node. The result is a network in which each node is an individual phone. The phonological rules are then repeatedly applied to the network. For each phonological rule the entire network is searched to find any nodes which satisfy the context conditions of the rule. Each rule provides an alternate pronunciation of some sequence of phones. If the alternate pronunciation is not already represented then an extra branch is created in the network representing the sequence of phones for the alternate pronunciation. This process applies across word boundaries as well as within words, depending on the phonological rule. Conditional probabilities for the different branches of the phonetic network are estimated from frequency of occurence statistics for a set of hand transcribed sentences. Such probabilities could even be made dialect dependent or even talker dependent. Note that the training sentences only need to be phonetically transcribed, it is not necessary to know the time at which each phone occurs since at this level we are no longer dealing directly with acoustics.

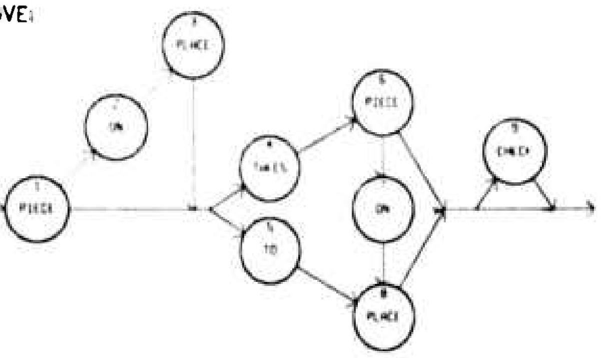
The explicit representation of phonological rules in the network is easily achieved at an expense of doubling or tripling the number of nooes in the network. However, with this stochastic network model it is not essential that an exhaustive set of phonological rules be used. In tact, implementations of the DRAGON system have been made with no explicit phonological rules and only one canonical pronunciation for each word. The reason that this representation is possible is that any phonological phenomena which are not introduced explicitly will be treated at the acoustic-phonetic level. Thus phonological substitutions can be mimiced by adjusting the probabilities in the matrix B[p,k] to include the probability that p is not the actual phone used by the talker but rather that some other phone q is spoken. Similarly, phonological insertions and deletions can be treated by adjusting the probabilities in the matrix $E[p_1,p_2,n]$. The disadvantage of this approach is that the matrices B and E represent less context than is available in the explicit representation of the phonological rules.

There is a serendipitous benefit in using the matrices B and E to represent acoustic-phonetic knowledge independently from the representation of the phonological rules. It the matrices B and E are estimated by running the acoustic preprocessor on a collection of test utterances, then any phonological rules which are left out in the prepared labeling of the test utterances are automatically absorbed into the estimates of B and E. Thus a perfect hand-labeled transcription of the test utterances is not only unnecessary, but undesirable. The best labeling for training purposes is an automatically generated labeling from a procedure knowing the sequence of words and having exactly the same lexical knowledge and phonological rules as the speech understanding system.

Representation of Syntactic and Semantic Knowledge

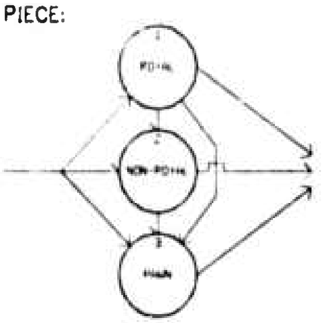
The syntax and semantics of a specific task domain can be represented by a multi-level network corresponding to a Markov process. Consider as a task a spoken chess move. Chess has a specialized grammar as well as a specialized vocabulary([6],[7]). Leaving aside a tew special moves, a move can be represented by a path through the following network:

MOVE:

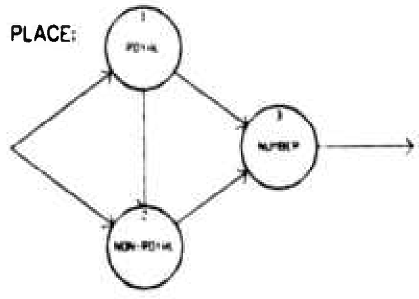


The nodes in the above network are not in general individual words, but are subgrammars which are themselves represented by networks. For example:

PIECE:

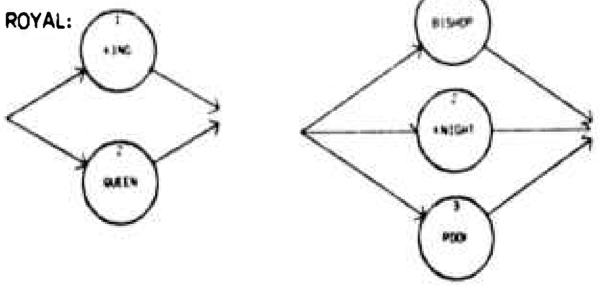


PLACE:



Again, the nodes can be expanded as networks:

ROYAL: NON-ROYAL:



It is clear that any regular (finite state) grammar can be represented by a tinite network. But in a speech understanding system the distinction between a regular grammar and an arbitrary context-dependent grammar is somewhat artificial. Consider the language of utterances generated by a particular grammar, not the sequence of words but the sequence of acoustic events. It is not unreasonable to assume, for example, that each entry in B[p,k] is non-zero, although perhaps very small. Such a result would automatically be the case, for example, if the conditional probability distributions for the acoustic parameters are multivariate normal distributions.

But if each entry in B[p,k] is non-zero, then at the acoustic level the language must include all possible sequences. Such a language can, of course, be represented by a tinite network grammar. Thus the issue becomes not one of generating the proper language, but rather one of modeling as accurately as possible the conditional probabilities, which can be context-dependent even for a context-tree grammar. Context is represented in the network by having separate nodes tor subgrammars which differ only with respect to context. For example, in the chess grammar there are two nodes marked "piece," one describing the piece which is moving and one describing a piece which is captured. There is clearly a trade off between the size of the state space and the amount of context which can be represented. For specialized tasks it is not difficult to achieve a reasonable representation of the grammar using most words at no more than two or three nodes. The transition probabilities for the grammar network can be estimated from statistics for a set of training sentences. A large set of training sentences should be used, but they only need to be transcribed orthographically, not phonetically, at this level of the hierarchy. If Bayesian statistics are used, the a priori probabilities could be set to achieve the same effect as a non-probabilistic use of the grammar. The a posteriori probabilities would then be a strict improvement (as judged by the training sentences).

To the extent to which the statistics of the training sentences reflect the true probabilities for spontaneous utterances for the specific task, the probability network represents not only the syntax of the task but also all of the recognition information which can be obtained from the semantics of the available context. That is, assuming the probabilities are correct, the probability network is an optimal predictor for a given amount of context, and therefore predicts at least as well as a human who is given the same amount of context and who presumably understands the sentence (although the context in this case is not the whole sentence).

Inter-sentence semantics can also be introduced into the probability network. One way to use inter-sentence semantics is to employ a user model. Suppose there is a model for the user in a particular task which gives probabilities for the user transitioning among a finite number of states depending on the types of utterances which the user has made in the past. Conceptually this model fits in easily as an extra level in the Markov hierarchy. Computationally it requires that conditional probabilities be estimated separately for each user state. However, since the user transitions between states only between utterances, a given utterance is analyzed using only a single representation of the probability network. The probabilities in this single network are weighted averages of the probabilities for the various user states. A user model is especially valuable if certain key sentences trigger user state transitions with probability one and if for each user state a small subset of the general grammar is used. Then there is a savings in both computation and storage requirements.

PERFORMANCE RESULTS

The testing of the system is still at too preliminary a stage to make any definitive conclusions, but initial results are very promising. Simulation studies have shown that the system can perform well despite a high error rate in the acoustic preprocessor. In its first test with live speech input, the system correctly recognized every word in all nine sentences in the test.

ACKNOWLEDGEMENTS

I wish to thank Leonard Baum, who introduced me to the theory of a probabilistic function of a Markov process, and Raj Reddy, whose encouragement, sponsorship, and personal examples have been my guiding light during this research.

BIBLIOGRAPHY

[1] Bellman, Richard Ernest, Dynamic Programming, Princeton University Press, 1957.

[2] Baum, Leonard E. and J. A. Eagon, "An Inequality with Applications to Statistical Estimations for Probalistic Functions of Markov Processes and to a Model for Ecology," American Mathematical Society Bulletin, vol. 73, (1967), pp. 360-362.

[3] Markov, A. A., "Essai d'une recherche statistique sur le texte du roman 'Eugene Onegin' illustrant la liaison des epreuve en chaine," Bulletin de l'Academie Imperiale des Sciences de St. Petersbourg, VII, 1913.

[4] Newell, Allen, et. al., "Speech Understanding Systems: Final Report of a Study Group," Computer Science Department, Carnegie-Mellon University, 1971.

[5] Paul, J. E., et. al., "Automatic Recognition of Continuous Speech: Further Development of a Hierarchical Strategy," Electronics Research Division, Rockwell International, 1973.

[6] Reddy, D. Raj, "On the Use of Environmental, Syntactic, and Probabilistic Constraints in Vision and Speech," Computer Science Department, Stanford University, 1969.

[7] Reddy, D. Raj, lee Erman and Richard Neely, "A Model and System for Machine Recognition of Speech," Computer Science Department, Carnegie-Mellon University, 1972.

[8] Reddy, D. Raj, Lee Erman, Richard Neely, et. al., "Working Papers in Speech Recognition--I," Computer Science Department, Carnegie-Mellon University, 1972.

[9] Tappert, Charles C., et. al., "Automatic Recognition of Continuous Speech Utilizing Dynamic Segmentation, Dual Classification, Sequential Decoding, and Error Recovery," International Business Machines Corporation, 1971.

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

Parameter Independent Machine Segmentation and Labeling

H.G. Goldbe.g, D.R. Reddy, R. Suslick
Computer Science Department(1)
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

Simple schemes are presented for segmenting and labeling continuous speech which are independent of the acoustic parameters used as input. Central to this approach is the belief that simple, parameter-independent structure is desirable at this level of speech recognition: 1) for comparisons among the various parametric representations for speech, 2) to provide a benchmark for any other scheme purporting to be better in either segmentation or labeling, 3) to avoid encoding in the algorithms the limitations of a representation, 4) to allow for more automatic training and adjustment, and 5) to study schemes that permit efficient hardware realization.

The segmenter is based upon the idea that significant change in a parameter should be sufficient evidence for a boundary, and that this evidence can be collected and viewed as a sum of weighted votes. A two-stage threshold network collects the vote sum and locates boundaries at local maxima in the sum, thus allowing context to have an effect.

The labeler takes a well accepted viewpoint from pattern classification research -- that distance in the space of acoustic parameters is strongly related to similarity in acoustic nature.

Three sets of acoustic parameters are used as input to the two procedures: amplitude and zero-crossing counts from octave band-pass filters (ZCC), smoothed LPC derived spectrum envelopes (SPG), and the frequencies and amplitudes of the first five peaks in the SPG (FMT). A straightforward training process is undergone for each parametric representation. Results are presented for a set of utterances spoken by the same speaker as the training corpus. The results obtained compare with human performance in segmenting and labeling with no syntactic or semantic support.

INTRODUCTION

Attempts at computer recognition of continuous speech have clearly pointed out the need for methods for dividing the speech signal into discrete acoustic segments and for labeling those segments in as accurate and robust a manner as possible. A number of specific methods for segmenting based upon particular acoustic parameters have been proposed. (see for example, Fant61, Reddy66, Denes68, Broad72) We believe that simple, uniform kinds of algorithms may be applied to the problem of segmentation and labeling of continuous speech in a manner independent of the choice of parametric representation of the speech signal. Although they may, doubtless, be improved by application of specific knowledge about the response of the parameters to particular speech phenomena, this knowledge has not yet been codified, or even aquired in sufficient breadth to support comparisons among the representations. The possible variations upon the methods for extracting parameters from the acoustic signal are endless, so it is imperative that a reasonably effective way of employing any such representation be found.

We will propose two such algorithms as benchmarks. We do not expect them to perform as well as more heuristic methods with significant amounts of speech knowledge, but they will provide as good an input to the higher levels of speech recognition as is found in many earlier systems and may be used as an off-the-shelf package. In addition, any method that proposes to advance the state of the art should do significantly better than these schemes.

There can be strong interaction between the segmenter and labeler. Information about segment identities may be used to verify or correct boundaries, on the other hand, the association of the input within a segment as all contributing to a single sound provides extra information to the labeling process. In the overall recognition system, these two processes combine to form a source of knowledge that transforms the acoustic signal into a sequence of discrete segmental phonetic identifiers. Later processing by higher levels may transform that sequence, correct it by applying rules of phonetic context, or even go back to the acoustic input in conditions that warrant more careful but expensive analysis. Primarily, we must deal with this level as a data reduction and transformation process.

Form of the Problem and Previous Methods

Most methods for analyzing the acoustic signal result in a vector of parameters at regular intervals in time. (1) The elements of this vector may be considered as measurements of features or as parts of an overall descriptor of the acoustic state of the signal. A great deal of effort has gone into the search for a set of such parametric measurements that display useful properties -- complete information (as verified by human perception experiments), orthogonality (or independence -- for better data compaction), independence of variations in speaker and equipment characteristics, etc. It was hoped that such parametric representations would lend themselves to the least errorful possible labeling of the phonetic content of the signal.

We are concerned here with the actual use of these parametric representations of the signal. We have a number of goals other than that of improving accuracy. They stem directly from deficiencies we feel are present in the current approaches at this level. Previous approaches have been ad hoc in their development. Typically, a representation is studied for its acoustic properties and the information obtained is codified in specialized rules. Even application of standard pattern classification methods is adapted to the particular pattern space by heuristic selection of weights and of classes based upon the strengths and limitations of the representation. Thus, there is no clear distinction between the efficacy of an algorithm for labeling or segmenting and that of the particular acoustic parameters.

1) There are rarely any comparative studies available because of the dependence of each system upon a priori assumptions. We would like to be able to perform comparisons among the parametric representations, the results of which we are

----

(1) This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research

(1) We also adopt the convention of vectors at uniform intervals. However, other methods (Baker74) show promise precisely because they do not average measurements over short time intervals, but rather measure specific events in time and the intervals between them.

confident can be extended to more heuristically coded production versions of the segmentation and labeling processes.

2) We would like to present a benchmark to the community, with enough performance capability to support a reasonable recognition system, but which must be surpassed if the other goals discussed here are sacrificed.

3) Many sets of parameters are correlated in well understood ways with one another, such as amplitude measurements in filter bands. In dealing with filter arrays, for example, one often implicitly encodes the concept of closeness in frequency with closeness in the array. We do not want to encode the structure, and the limitations, of a particular parametric representation into the algorithms unless we are satisfied that the advantages of doing so outweigh the loss of generality and flexibility. While there is nothing wrong and much to be gained in using this information -- the best systems will have to, we would like to have some confidence in our choice of parametric representation before we do so.

4) As well as comparative rating of parametric representations and benchmarks, there is also a need for methods that are straightforward in structure and implementation. Available schemes for unsupervised learning and for tracking of slowly or infrequently shifting clusters in the pattern space depend for their success upon an uncomplicated model of pattern classes and uniform treatment of the dimensions of the pattern space.

5) Such algorithms are more easily realized in hardware, with the consequent speedup so available. Since the algorithms are designed to be independent of the particular acoustic parameters, fixing them in hardware will not be as big a risk as one might think.

Other problems arise in dealing with variations introduced by different speaker and equipment characteristics, or different vocabularies and hence phonetic contexts. Their effects, while significant to the operation of a complete speech recognition system, are secondary in this context. We expect that the results obtained over uniform, high-quality data, with the simple algorithms we are proposing here, will degrade gracefully with the introduction of other sources of variation and noise into the data.

SEGMENTATION

The first process we would like to apply to the input data is to segment it in time into related acoustic segments. This is often done at a later stage, after some labeling, or at least recognition of features such as "voiced", "fricated", or "silence" has been attempted at small regular intervals. However, our approach is to attempt the segmentation initially, in order to have that segmentation as useful input to the labeling process. If we err in favor of too many boundaries, we may always combine segments with similar labels, once those labels are placed.

Evidence for Segment Boundary

Clearly, the concept of acoustic similarity and difference is central to any segmentation procedure(1) Thus, one might, instead of labeling each interval, label the interstices between the intervals, i.e. measure the difference, according to some classifying rule, between adjacent intervals. The distance, in some parameter space for example, between the pattern associated with a noise-like interval (fricative) and that of a nasal-like interval would be great enough to signal the placement of a boundary, while the distance between patterns for high and middle vowel-like sounds might not and probably should not. There is definitly an element of risk in adopting such a decision strategy -- that important boundaries will be missed because:

1) the distance measurement is not sensitive to change in certain directions in the acoustic space,
2) the parameters do not reflect such changes,
3) the change is too slow,
or 4) the magnitudes of the changes vary considerably with context,

and thus not be susceptible to easy decision rules. Problems 1 and 2 will bother any segmentation procedure, and must be solved by choosing better parametric representations for speech. The problem of slow change, 3, will also plague many different algorithms. It is a peculiarity of speech that must be dealt with. Problem 4, varying magnitude of change, can be approached fairly simply by treating the change as a signal in time. We will show one possible approach.

The rule we have chosen is based upon the idea that each parameter of the speech signal can be viewed as a separate channel of time varying information about the utterance. (Figure 1) A sudden or significant change in even a few channels should signal a boundary. Thus we may collect evidence about the placement of boundaries by placing a threshold on the change in each channel, and report when the threshold is exceeded over adjacent time interval differences. Because we expect some changes to occur gradually, we measure the change between intervals one unit further away (a total of three units) as well and allow them to react to another set of thresholds.

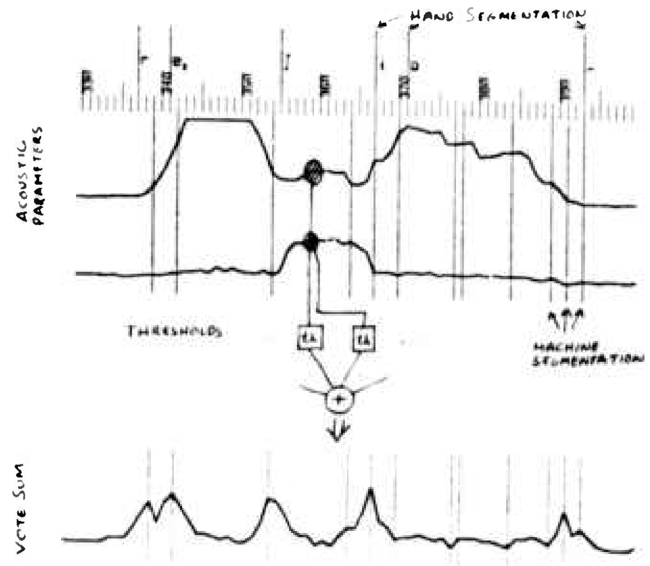


Figure 1 -- Segmenter Voting

A second stage is needed to combine these votes for change of the individual threshold units. It this were a Perceptron recognizer, this second stage would not have memory and could not take context in time into account except as it was explicitly measured by a weighted combination of the primary stage units. However, this will not work when changes vary in magnitude and the number of channels affected. The acoustic context greatly affects the suddenness and severity of a boundary. There is no threshold that can be used on the vote sum, for example, an overall vote level that specified change from fricative to vowel would be too large to work for silence-nasal transitions where only a few parameters may change. The situation we wish to recognize is that, whatever change does occur, it is greatest at the point we wish to mark. Thus a local maximum is found in the sum of the threshold unit votes. These votes are weighted to emphasize the adjacent interval

----

(1) If each interval is labeled as some acoustic type, the grouping together of strings of these labels, as is often done, according to higher or broader type classification is just an assertion that boundaries should occur where adjacent intervals belong to very different acoustic types.

differences over the longer slower changes. However, the actual differences are not summed. Rather, a vote for change is considered to be of equal importance from any channel if it triggers over that channel's threshold.

Transition Segments

The local search for maxima can also incorporate a measurment of slope or area to try to characterize the gradualness of change. Broader peaks in the vote sum wil indicate transitionary portions of the signal which are changing acoustically over a longer time than usual. We have had some succes in distinguishing such segments by measuring the width of the vote sum at a given drop below each peak. If the width exceeds some preset limit, the peak is considered to represent, not a boundary, but a transitionary segment and is marked accordingly.(1) Difficulties occur because: i)Some such segments are transitionary only in one channel or only slightly as compared to the entire signal. Hence the vote sum itself is very low and the width measurement is confused by noise effects from other channels. ii) Noise in the signal or parameter measurements may give the effect of transitionary segments by masking a sharp change. While the method described here has not yielded what we would consider good identification of transitionary segments, it has improved the location of boundaries. This suggests that we may not have a clear idea of what kind of phenomenon we mean by "transition." (See Figure 2 for some examples.)

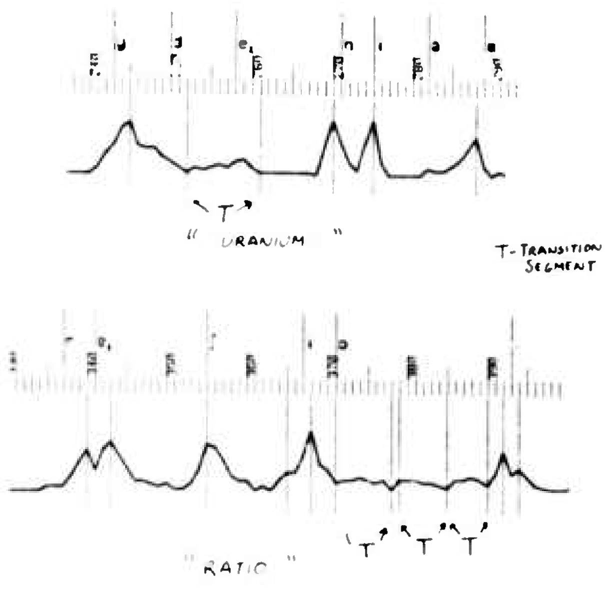


Figure 2 -- Vote Sum Peak and Transition Location

Training

Finally, we must specify the threshold values to be used for the primary stage voting. These will, of necessity, depend upon the parametric representation chosen, but will depend in a uniform manner upon it. By uniform we imply that a standard procedure for training will be sufficient and may be applied without a great deal of knowledge about the parameter space. We have obtained good results by setting all the primary stage thresholds to the same value. The results then depend upon that one value and a significance threshold used for ignoring small peaks in the vote sum. The other parameters of the process are the weights of the threshold unit votes. We have argued that they ought to be the sam over all channels, and have fixed them at two and one for the one and three interval differences respectively. However, the relative importance of the channels may be learned from the training data fairly easily. These few parameters form a small set of values through which one may search with a corpus of hand marked data.

A more direct method for learning the thresholds is to collect the values of the differences at hand marked boundaries on a training set of utterances. At each boundary, the largest difference for each time span (1 or 3 for example) is considered relevent and is used to force the threshold down to its value. A preliminary look at histograms of these differences (Figure 3) will show a level below which one should ignore the boundary as spurious. (Often, hand labeled boundaries do not occur at points of any acoustic change, but represent the segmenter's idea of a phonemic boundary.) The resultant thresholds should be able to recognize at least all the non-spurious hand marked boundaries, and probably will mark more. This tendency towards to many boundaries is, we feel, the least of many evils. The frequencies with which a channel show the greatest change will give a good idea of relative importance for voting weights, if they are desired.

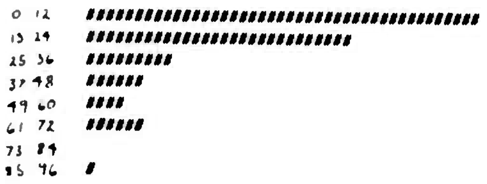


Figure 3 -- Histogram of Differences at Hand Marked Boundaries
2nd Formant Amplitude Parameter

LABELING

A great many algorithms have been proposed for labeling a piece of acoustic data with its phonetic type. Although this problem seems to fit directly into the basic pattern classification model, and although pattern classification research has developed methods for a variety of situations, the general consensus has been that these methods are not sufficiently powerful to solve the speech recognition problem. We feel that this is a negative reaction to initial failures. Even though the identification of phonemes by uniform classification rules will probably not be accomplished -- there is no reasonable representation of the acoustic signal that contains all the needed information about context, prosodics, and coarticulation to allow classification at the phonemic level -- the methods developed for classifying vector patterns can be successful if greater effort is spent in choosing the classifier, aquiring the relevent statistics, and choosing the proper classes for speech.(1)

Distance in Pattern Space

One important way of viewing a pattern to be recognized is that it is represented by some point in a space of possible patterns, and central to that conceptualization is the notion that the distance between two points in the pattern space relates to similarity of the patterns represented by them. We have compared a number of distance measures that are well known to pattern classification research and have chosen a few simple distances that essentially provide linear classifying boundaries in the pattern

----

(1) Other attempts have treated every boundary as a (possibly) short transition segment. (Reddy66)

----

(1) This last issue involves a greater understanding of the statistical nature of the pattern space than is available. How do the clusters relate to one another, what are the significant subclasses of a phone, and how will the label be used in the rest of the recognition system?

space. They are correlation (the n-space angle), Euclidean distance (the magnitude of the difference vector), and Euclidean distance in a variance normalized space.(1) (See figure 4)

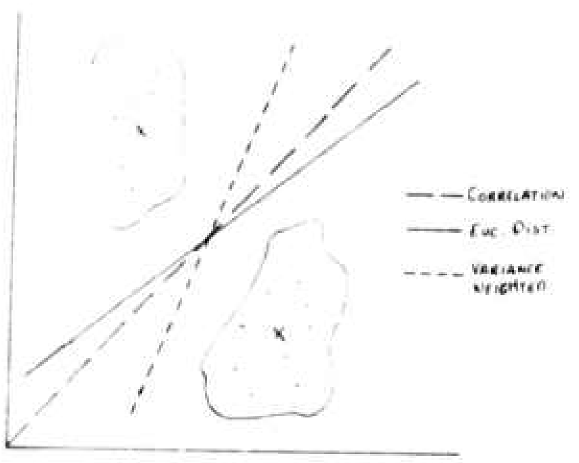


Figure 4 -- Decision Boundaries

More complex partitioning of the pattern space can take many forms. (Nagy68, Meisel72) Often some estimation of the density function of the patterns within each class is made, then a Bayes optimal rule is defined by choosing the class with the greatest a posteriori probability. However, the computational requirements of such a calculation, the difficulty of estimating the densities in question, and the fact that they will have to be easily altered as conditions and speakers vary suggest that simpler methods be used.

The algorithm is simply to compute the "distance" between the unknown pattern and each of the clusters in turn. The clusters are defined by whatever statistics they may require, such as the mean and standard deviation of each element over the training samples. Although this requires more computation than a successive splitting of the space into subsets of classes, it is more flexible and does not require a hierarchy of classes.

When the classes to be recognized are composed of multiple sub-classes that are themselves more well defined (more tightly clustered in the pattern space), a good approach is to form partitions that separate the sub-classes and then combine them by rule. This is sometimes called drawing a piece-wise boundary, and is closely related to the nearest-neighbor and Parzen window methods.(2) There are well accepted hierarchical divisions of the speech sounds that may be used to provide such a sub-division. We have chosen to define simple clusters that correspond to a set of 77 most significant allophones of English(3) (Table 1). These become labels that may take on different acoustic and phonetic meaning as training progresses -- they lose their phonemic meaning except that we begin by collecting statistical information about these allophones from a hand labeled corpus of data.

----

(1) The elements of the pattern vector are normalized by a weight proportional to the standard deviation of elements of the patterns in the particular class in question.

(2) One level of the Stanford signature tables is devoted to building up a piecewise description of the pattern cluster associated with a phone class.

(3) Shockey, L., Private communication, January 1974.

-					(Silence)
B	D	G	DX	Q	(Silence-like)
P	T	K	F	TH	(Soft fricative or burst)
V	DH	Z	ZH		(Voiced fricative)
HH	WH	S	SH		(Fricative)
Y10T	W10T	R10T	L10T		(Unvoiced glide)
Y	W	R	L	EL	(Glide)
M	N	NX	EM	EN	(Nasal)
IY	IH	UW	UH		(Vowel -- neutral)
EH	ER	AX	OW		
AO	AE	AA			
	.....,	(Vowel -- velarized, nasalized, retroflexed)			

Table 1 -- Phonetic Classes, Initial Definitions

The hand labeling process, of necessity, involves some interference by the concept of phoneme, although we have attempted to label sub-phonemically -- to label the separate sounds within a phoneme. Once the initial statistics are gathered, training procedes by applying the machine segmenter and labeler to the same corpus. This provides a second set of labeled data which may be used to compute new statistics. In the case of Euclidean distance from the cluster means, the process described can be shown to converge to a minimum intra-cluster scatter. In any case, after a few iterations the clusters have changed their character and can no longer be considered as allophones. They do, however, provide consistent labeling of a wide variety of acoustic phenomena, and the phonetic correlates of those labels can be seen in an inspection of the training corpus and what class labels occur in various phonetic contexts.

Multiple Labels - the Entire Segment and Classes for Labeling

To enable the labeler to use information from the segmentation, we keep an ordered list of the best few labels (usually 5) for each time interval (each pattern vector) in a segment's center half. These contribute to selection of the segment label by voting with a weight determined by their position in the ordered list. We have had reasonable results from the weights, 5,4,..., but there is clearly room here for application of better information. An estimate of the a posteriori probability of the label could be made from the values returned by the distance measure, for example. This voting scheme additionally provides an ordered list of labels for the segment. We have chosen to output the entire list for use in higher level analysis, since often the top two or three labels are close in their scores. The rules for extracting phonetic features from these sets of labels are being developed as a source of knowledge for the Hearsay II system at Carnegie-Mellon University. (Lesser74) The use of these labels becomes an interesting problem. They clearly have acoustic meaning, since that is defined by the cluster statistics and the classifier rule -- i.e. by a piece of the pattern space. However, they have phonetic meaning as well, because they are interpreted in the light of the phones (from a hand labeled corpus) within which they occur. This acoustic-phonetic correlation can be quantified and modeled by the frequencies of the abovementioned occurances. If the frequencies are treated as probabilities that a label will be realized within a segment corresponding to a particular phone, Bayes rule can be used to estimate the a posteriori probability that the phone was there. (1)

----

(1) This has not yet produced good results -- possibly because the numbers used to model the phone to label probabilities are not very good. An important element in the Bayes calculation is the a priori probability of each phone. This might be supplied by the higher levels from analysis of sequences, hypothesized words, probable length of segments, etc.

RESULTS and CONCLUSIONS

The results obtained with the uniform algorithms we have presented should be considered in the light of their usefulness to a larger system. We recognize that these methods are weak in comparison to what humans can do and to what we will need for successful recognition of continuous speech with relatively unconstrained syntax and semantics. However, Shockey and Reddy (Shockey74) measured accuracy of phoneme identification by humans working from spectrograms, from waveforms, and acoustically in foreign language utterances where no higher level support was available. The results they obtained may put bounds on our reasonable expectations of machine recognizers. As one would expect, acoustic input provided a much better identification rate than the graphical representation, which were about equal. Yet the actual rates were approximately 30% (waveform or spectrogram) and 70% (acoustic) for a set of about 50 phonemes. This would indicate a considerable reliance upon higher level processing is necessary. Identification into about six, gross types occured with rates of 80% and 95%. We suggest that a machine recognizer at the local classification level of a system would be doing well to provide recognition in the 30/80 range until more is made available about the particular mechanisms that enable humans to process acoustic information.

Table 2 summarizes the results of the algorithms on three different parametric representations for a corpus of five sentences:

What is the average uranium lead ratio for the lunar samples?
Do any samples contain troilite?
Who is the owner of utterance eight?
Where were you when we were all away?
We all heard a yellow lion roar.

The three representations were:

ZCC -- 12 parameters, Amplitude and Zero-crossing count from each of 5 octave filter bands and unfiltered

SPG -- 128 smoothed spectral envelope points from LPC coefficients (Markel68)

FMT -- Formant frequencies and amplitudes from the SPG envelope, 10 parameters

The labeling distance measure used was Euclidean distance weighted by the variance. The segmenting thresholds were all obtained by the training method discussed earlier. The utterances were recorded under the same conditions and by the same male speaker as the corpus of utterances used to gather statistics for the labeler, to train the segmenter thresholds, and to refine the cluster set. However, we have observed only a mild reduction in accuracy when data recorded by other male speakers is analyzed.

	ZCC	SPG	FMT
Labeling (percent correct)			
Exact Label	14	32	8
Rough label	69	79	47
Segmenting (number - out of 134 hand marked segments)			
Missing	13	3	6
Extra	59	138	112

Table 2 -- Results of Machine Segmenting and Labeling

Remarks: 1) The counts of missing and extra segment boundaries are highly negatively correlated, thus the high number of extra segments which SPG and FMT display explains their low missing segment score. This was due primarily to poor training of the tresholds. However, the extra segments were usually labeled properly and could easily be recombined.
2) The rough label score is the percentage identified into the correct class of about 10 broad classes of speech sounds. This was done to compare with the foreign language experiment refered to above.

Errors in Segmentation

We can separate segmentation errors into three types: errors of extra segments, missing segments, and transition indentification. The probable effect of an error upon a speech understanding system and, specifically, the labeling process, will vary considerably with the type of error.

Extra segments -- We have biased the threshold training towards thresholds that will produce too many segment boundaries. These errors are not very serious since, if a sequence of short segments are labeled with similar labels that indicate a sustained-type phonetic situation, then the segments may be combined and the labels collected by a voting scheme similar to the one used to combine individual intervals. The most common occurance of this phenomenon is during silence segments. The other common situation is during gradually changing sustained segments, usually trailing off into silence at the end of a phrase. These may also be detected by the characteristic short segments with related labels.

Missing segments -- This is a more serious type of error since it requires, for correction, that the rest of the system hypothesize the existence of the missing segment. In addition, it causes the labeler to combine information from two segments that are acoustically similar, but do differ somewhat. Very often, the errors that seem to be of this type are actually indications of a case where a phoneme boundary "exists" but no phonetic change occurs. Manual segmentations often contain such boundaries, and we must relie upon the higher levels of analysis to postulate such non-acoustic divisions of the utterance. Most of the significant problems seem to occur at glide-vowel junctures. This appears to us to be the kind of problem that can be dealt with after some initial labeling has occured. If we have located a sonorant segment with glide and vowel characteristics, we may invoke a formant tracker, or a specialized segmenter that understands the parameter space as it relates to the classes in question. It may make considerable demands upon system resources, because it is only used when needed.

Transition identification -- It is reasonable to treat every change from one sustained segment to another as a transition segment. We have attempted to identify only those transitions which occur for a significant length of time. Since this is a subjective quality, there can be no absolute measurement of correctness. What we have observed is that the transition finding process seems to help in some cases where boundaries should be located at the beginning or end of change rather than at the point of greatest change, and it does not hurt in most other cases. Clearly, more accurate transition identification could be done using the labeler output at a higher level in the system.

Errors in Labeling

The errors encountered in our attempt to do phonetic labeling will be less critical if there is information available to the speech understanding system to correct those errors when other constraints indicate that the initial label choice is wrong. We presented a simple way of providing this information by returning the top few labels as they were rated over the center half of the segment. The approach in Hearsay II (Lesser74) will be to extract features from this list of labels, however, other uses could be made as well. One should consider a labeling algorithm good if the correct phonetic label occurs in the top few, and especially if it is strongly reinforced by phonetically similar labels.

Some labeling errors occur because the segmenter has failed to separate two different segments. Usually some characteristics of each can be seen in the labels, but the confusion can be serious. Most errors, however, are direct results of the inadequacy of the parameters to represent the acoustic "difference" as a simple distance. Goldberg performed preliminary rating of some parametric representations and simple distance measures (Goidberg73). The results are not unexpected -- spectral envelopes did fairly well, for example, as did a generalized quadratic classifier based on assumptions of normality. The interesting point is that the best results fell into the range of human performance shown by Shockey and Reddy.

Conclusions

We have shown that the same uniform algorithms may be used to nroduce segmentation and labeling from quite different parametric representations of the speech signals. The ability to make comparisons is thus made available. The algorithms are simple in form, and thus easily implemented in hardware. Their performance, while not at the state of the art, is not far behind it. We would recommend a "front end" of such methods for a straightforward speech system. Such systems will be desired to test new ideas for higher levels, to provide man-machine communication in highly constrained tasks, and to test basic changes in system structure.

Our plans include the application of these algorithms to other parametric representations than we have presented here. A comparitive evaluation is being made of a variety of parametric representations for their ability to support segmentation and labeling.

BIBLIOGRAPHY

[Baker74] Baker, J.M., "A New Time-Domain Analysis of Fricatives and Stop Consonants," IEEE Sym. Speech Recog., Carnegie-Mellon University, April, 1974 (this volume).

[Broad72] Broad, D.J., "Formants in Automatic Speech Recognition," Proc. Int. Conf. Speech Commun. Processing, 1972.

[Denes68] Denes, P.B., von Keller, T.G., "Articulatory Segmentation for Automatic Recognition of Speech," Proc. 6th Int. Congr. Acoust., Vol. B, 1968.

[Lesser74] Lesser, V.R., Fenneil, R.D., Erman, L.D., and Reddy, D.R., "Organization of the Hearsay II Speech Understanding System," IEEE Sym. Speech Recog., C-MU, April, 1974 (this volume).

[Fant61] Fant, C.G.M., Lindblom, B., "Studies of Minimal Speech Sound Units," Speech Transmission Lab, Quarterly Prog. Stat. Rep., Vol. 2, 1961.

[Goldberg73] Goldberg, H.G., "A Comparison of Parametric Representations for Speech," Unpublished working paper, C-MU, Pittsburgh, June 1973.

[Meisel72] Meisel, W.S., "Computer Oriented Approaches to Pattern Recognition," Academic Press, N.Y., 1972.

[Nagy68] Nagy, G., "State of the Art in Pattern Recognition," Proc. IEEE, Vol. 56, No. 5, May, 1968.

[Reddy66] Reddy, D.R., "Segmentation of Speech Sounds," J. Acoust. Soc. America, Vol. 40, No. 2, August, 1966.

[Shockey74] Shockey, L., Reddy D.R., "Human Segmentation of Unfamiliar Speech," to be presented Acous. Soc. Am. meeting, N.Y., April, 1974.

A NEW TIME-DOMAIN ANALYSIS OF FRICATIVES AND STOP CONSONANTS*

Janet MacIver Baker
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

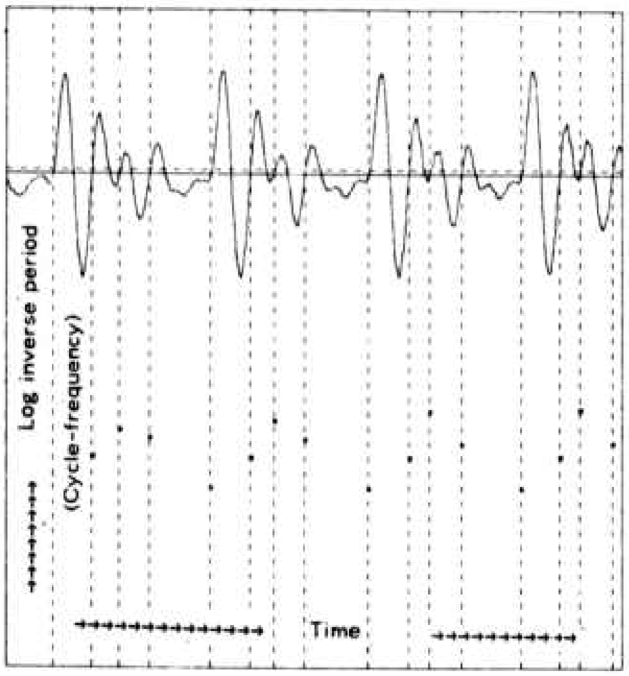
Time-domain analysis has proven quite useful for revealing meaningful acoustic transients in human speech. Although many of these transients are both quite brief in duration and low in amplitude, they occur consistently in connected speech. This paper outlines the kinds of analyses performed and their results pertaining to the fricatives and stop consonants.

←←←←←←←←←←←←→→→→→→→→→→

This paper describes the results of applying our new time-domain techniques to the analysis of complex waveforms, in this case human speech. Their chief advantage is precise temporal resolution allowing exact timing of articulatory events within a sample of speech; that is, no bandwidth limitation is present. This temporal resolution is most significant for characterizing fast transitional regions such as occur at vowel-consonant and consonant-vowel boundaries and within stop consonants. In addition, certain characteristics of these regions are either greatly enhanced or uniquely apparent in the time-domain. Such information is revealed in our visual displays generated from the speech waveform up-crossings in time. We call these log inverse period (LIP) plots.

The impetus for this work comes from two sources: 1)First are the studies by Licklider and his colleagues (5,6,) who 25 years ago demonstrated the intelligibility of infinitely clipped speech. This showed that sufficient acoustic speech information is encoded in the zero-crossings of the waveform itself. Given the redundancy of speech such information is most probably encoded by other aspects of the waveform. As it happens though, zero-crossings or up-crossings are easy to see and extract from the waveform. 2)The second motivation for this work comes from neurophysiological research on the auditory information processing of the ear itself. Basically the ear processes an incoming signal in at least two widely recognized manners. The first is analysis in the frequency-domain and is analgous to a kind of filter bank where different neurons along the basilar membrane respond to different frequency ranges; that is, a given neuron fires if it detects a signal of sufficient intensity within a particular frequency range. Neurons also code information in the time-domain in a manner known as phase-locking (4,8). Given a signal waveform, a phase-locking neuron responds by firing once, phase consistently, for each cycle or integer number of cycles within the waveform. The technique we are using is directly analagous to this latter time-domain coding technique.

We generate our visual displays as follows: A zero-axis is drawn horizontally through the center of the acoustic waveform. We note the exact time when the waveform crosses this axis in an upward direction. In actuality, we usually record only those up-crossings which exceed some threshold amplitude, epsilon, set slightly above the horizontal zero-axis. This threshold tends to preclude low amplitude background noise. We measure each interval between successive up-crossings and plot these as a function of time in our displays. Therefore each up-crossing in the acoustic waveform is represented by a discrete dot in our displays. In fact, we actually plot on a log scale, the inverse of the interval between successive up-crossings, the period of the cycle, along the vertical y-axis and time along the horizontal x-axis. This yields a display which superficially resembles a kind of spectrographic display. (N.B. For those readers familiar with neurophysiological studies of single unit responses, this display is analagous to an "instantaneous frequency" plot and functionally analagous to a phase-locking phenomenon.) We also display a rough intensity measure by means of a z-axis modulation. That is, the size of a dot representing a given cycle is proportionate to the log of the greatest amplitude achieved during that cycle. This dot size intensity measure in our up-crossing displays is analagous to the intensity measure expressed in spectrograms. The following illustration shows the relationship of the log inverse period plot to the waveform from which it is generated. Note that individual cycle-frequency values may be easily read from the y-axis.



The idea of looking at zero-crossing measures per se is not in itself conceptually new. However, in contrast to most other investigators (2,3,7,9) who have used zero-crossing measures to analyze speech, we do not average our up-crossings over a fixed interval of time. Reasons for this will be discussed shortly. First of all it is important to be aware that the chief motivation for many zero-crossing studies has been in searching for an inexpensive way to find frequency domain acoustic features, such as formants. This method avoids the computations required for Fourier transforms, for example. In order to decrease the expense and variability in examining individual cycles, it was easy to to compute an average cycle length by simply counting the number of zero-crossings occurring during a given time interval. This procedure has two major consequences: 1) the perfect time resolution inherent in the time-domain is lost when crossings are averaged; that is, a bandwidth limitation is introduced, 2) the conventional acoustic features extracted are usually less precise and more variable than the same acoustic features extracted directly with a frequency-domain analysis. Ou

*This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

reason for not averaging up-crossings is that in the speech waveform itself there are significant acoustic features which last for only one or a few cycles in duration. If cycles are averaged, this information is irrevocably lost. Such transient events frequently occur at vowel-consonant and consonant-vowel boundaries as well as between other acoustically distinct regions, within stop consonants for example.

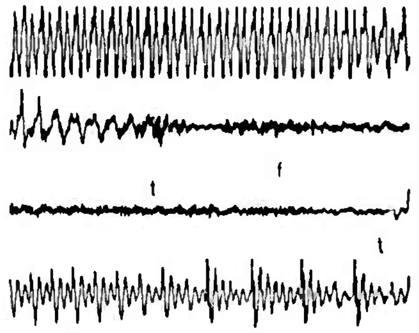
Data and Methods

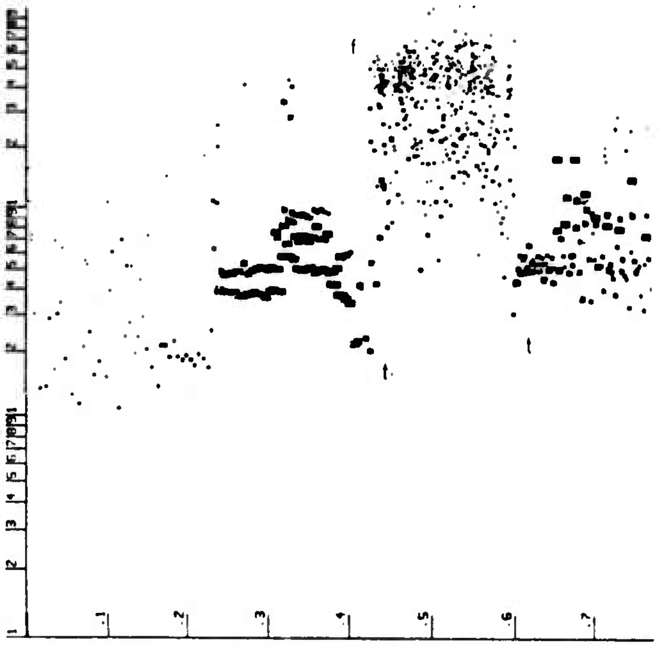
The total amount of data examined during the course of this investigation consisted of several thousand utterances, in both citation form as well as connected speech, spoken by more than 20 male and female speakers, often in noisy environments. The set of this data which has been studied most thoroughly consists of 684 utterances in citation form, generously provided by June Shoup. Each of the speakers (2 male and 1 female) spoke 228 utterances chosen designed to provide examples of all the allophones of the fricatives and stop consonants common in the English language, as described in June Shoup's Ph.D thesis, 1964 (10).

Each of these three sets of recordings was digitally sampled at 20 kHz. Then a number of time-domain measures were computed from these digital files. The accuracy of such measurements is of course limited by the 50 microsecond resolution of the sampling. However linear interpolation between two successive samples was routinely performed to more accurately pinpoint the time of waveform up-crossings. The time for each waveform upcrossing was computed and used to calculate the inverse period for each cycle in the waveform. Various amplitude measures were computed for each cycle as well as several measures of the amount of microstructure riding on each cycle. Each of these three types of time-domain parameters have proved to be quite useful. Then with all of these parameters available, a cycle-by-cycle hand analysis of the waveforms for all 684 utterances was performed in order to precisely mark the time at which sharp discontinuities in one or more of these parameters delineated the acoustically distinct segments which occur internally in fricative and stop consonants. This precise segmentation required correlation of the time-domain parameter values with the LIP plots and expanded waveforms. Statistics on each of these acoustic segments were then computed with respect to each of 18 linear and logarithmic time-domain parameters. In all there were 23 different statistical tests performed on the individual acoustic segments for each fricative and stop consonant. These tests included finding the number of cycles in the sample, the mean, maximum and minimum values, standard deviation, bimodal distribution etc. In addition, where values of individual cycles within a given segment were more than 2 standard deviations from the mean for the whole set, these cycles were eliminated and statistical measures, as described above, were computed for the remaining set of cycles. Also, for each segment a least squares linear fit was computed and its values at the beginning and end of the segment, respectively, were derived. These latter measures are particularly useful for indicating whether a given segment is relatively steady state and how great a discontinuity occurs at the end of one segment compared with the beginning of the next.

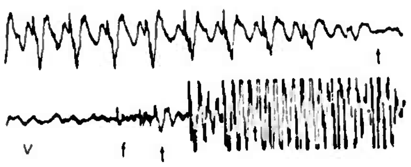
Fricatives

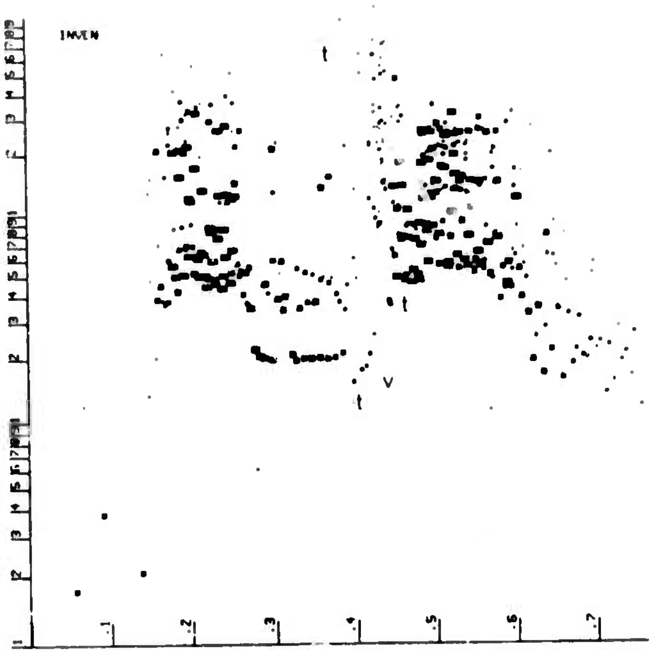
Generally fricatives are acoustically characterized as sustained high frequency regions. In voiced fricatives, this high frequency region is preceded by a low frequency region which may persist throughout the high frequency region as well. Time-domain analysis reveals that at the beginning of the high frequency portion of the fricative, there is a very sharp discontinuity simultaneously, upward, for both cycle-frequency and microstructure and often a decrease in amplitude where the fricative is preceded by a vowel. These are all large changes which are usually sustained for the duration of the fricative. Usually at the end of the fricative, sharp discontinuities are again observed. However a much more transient kind of acoustic feature often occurs at the very beginning and again at the end of the fricative. At these places is found one or a few cycles characterized by lower cycle-frequencies than those of the other cycles in the acoustic segment immediately preceding and the acoustic segment immediately following this transitional phenomena. Amplitude of these cycles is variable and cycle microstructure is usually low. These transition cycles are marked "t" in the LIP plots. Regions of frication are marked "f" and for voiced fricatives, the initial lower frequency region is marked "v". Each line of waveform represents .1 sec of the speech signal analyzed. Similarly, the x-axis of the LIP plots is marked at .1 sec intervals. The first example is an /s/ from the utterance "there sir" (female speaker, HN).

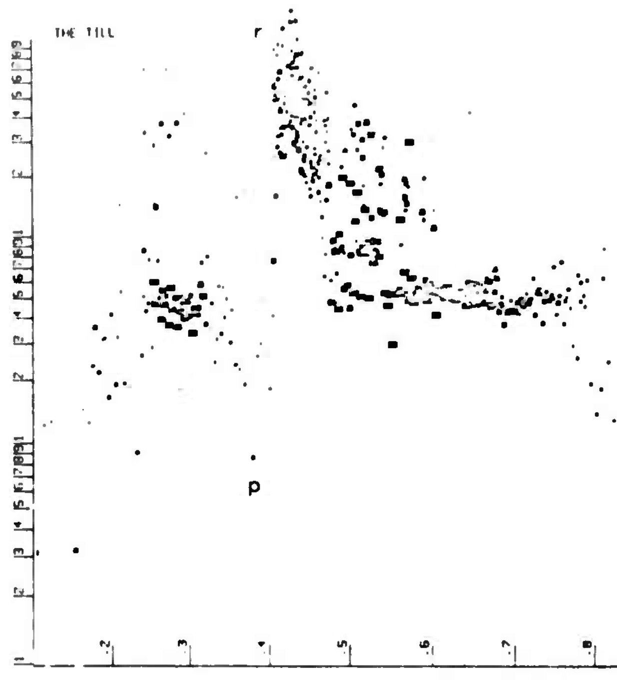




The second example shows the voiced fricative /v/ in the utterance "invent" (male speaker, EH).





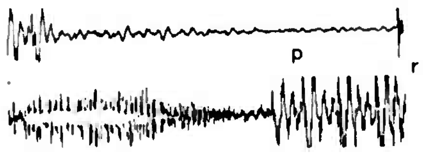


Stop Consonants

Acoustically, stop consonants usually have a pause portion followed by a higher frequency region which represents the stop consonant release region, plus aspiration if present. A voiced stop consonant has a low frequency or voicing region just preceding the pause portion. Often these lower freqencies are sustained throughout the release-aspiration region as well. And it is not uncommon for the pause cycles to be completely omitted in a voiced stop consonant.

As the waveform transitions from prior context or the initial voicing region in voiced stop consonants, the cycle-frequency, amplitude and microstructure drop sharply. Although this pause portion lasts only one or a few cycles, the cycle-frequencies are quite low, often less than 100 Hz. The dots representing these low cycles are visually quite obvious in the LIP plots. Next, as the waveform transitions abruptly into the release-aspiration region, both cycle-frequency and microstructure measures increase sharply as does amplitude, which nonetheless at its peak value generally remains well below the average level for unstressed vowels. Where aspiration is present, the transition from release to aspiration is often smooth with cycle-frequencies and amplitude gradually decreasing.

In the LIP plots shown here, pause cycles are marked "p", the release-aspiration region by "r", and the initial voiced region of voiced stop consonants by "v". The following example is of the /t/ in the utterance "the till" (male speaker, EH).



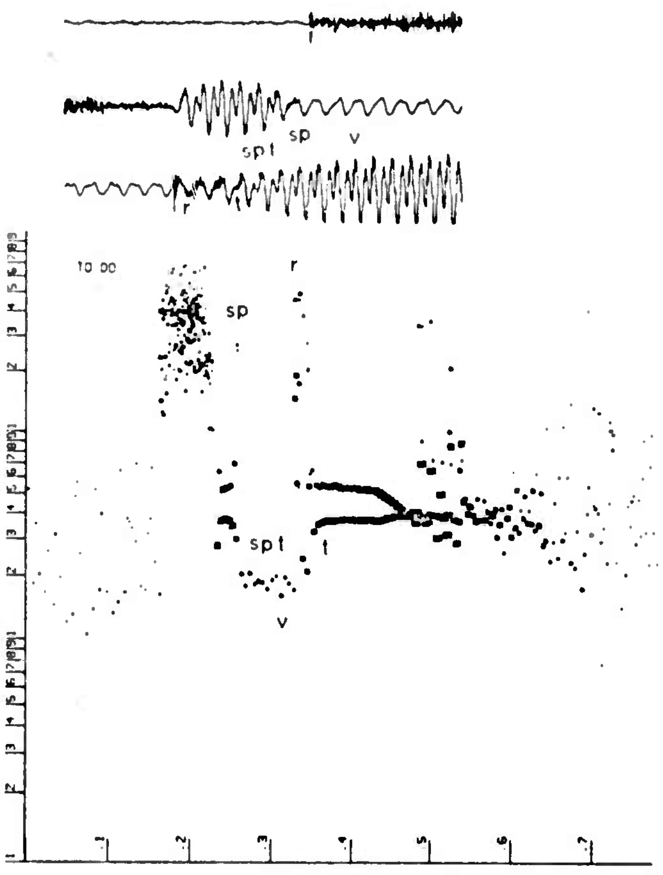
Time-domain analysis also reveals the existence of several more subtle acoustic phenomena which have previously gone unrecognized. These phenomena are often both short in duration and low in amplitude. They occur often at phone boundaries are last for only one or a few cycles in the acoustic waveform.

The first of these is analagous to the transitional cycles previously described for fricatives. At the end of the release-aspiration region of the stop consonant, there is often, though not always, one or a few cycles which have lower cycle-frequencies than any of the other cycles in either of the acoustic segments immediately preceding and following this acoustic event. These transitional cycles are marked as "t" in the LIP plots which follow.

The second phenomenon is very common and shall be referred to as a "stop preview". In the case of a stop consonant which is preceded by a vowel (and sometimes by other phone types as well), the very end of the vowel is acoustically characterized by one or two cycles with much higher cycle-frequencies than any of the other cycles which comprise the vowel. These stop preview cycles are very low in amplitude.Their duration is almost always less than 1 msec and very commonly less than .5 msec. In the LIP plots, these are marked as "sp".

The third phenomenon concerns the one or two cycles immediately preceding the stop preview. These one or two cycles are usually of relatively large amplitude, but have a lower cycle-frequency than those of the cycles preceding it. Only at the very beginning of the vowel are there cycles with cycle-frequencies as low or lower than the cycles immediately preceding the stop preview. Although these stop preview transitional cycles are sometimes omitted when the stop preview is present, they have not been observed when the stop preview itself is absent. They are marked as "spt" in the LIP plots.

Illustrative examples of all these phenomena are provided in the utterances 1) "to do" and 2) "he grows" (female speaker, HN).

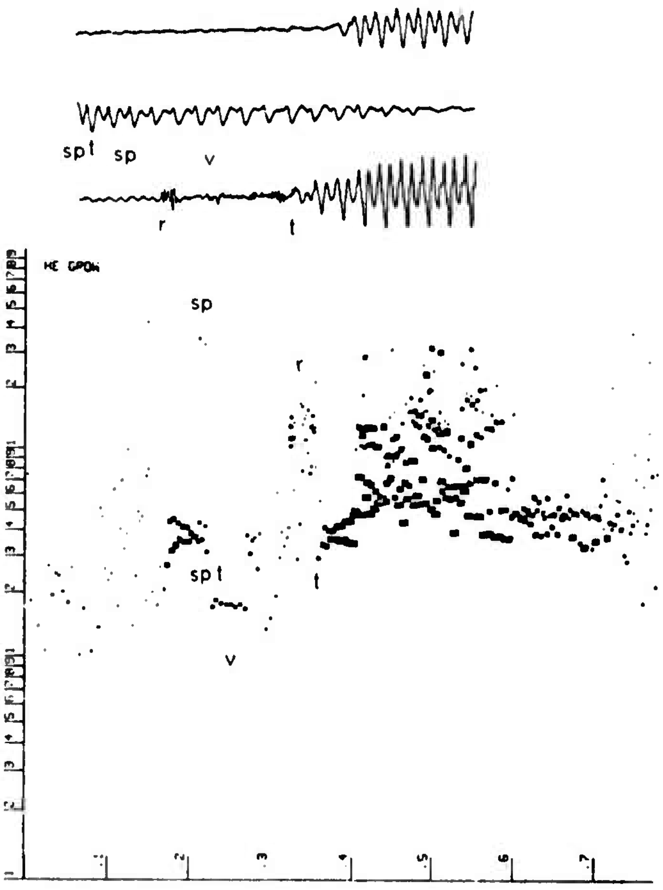


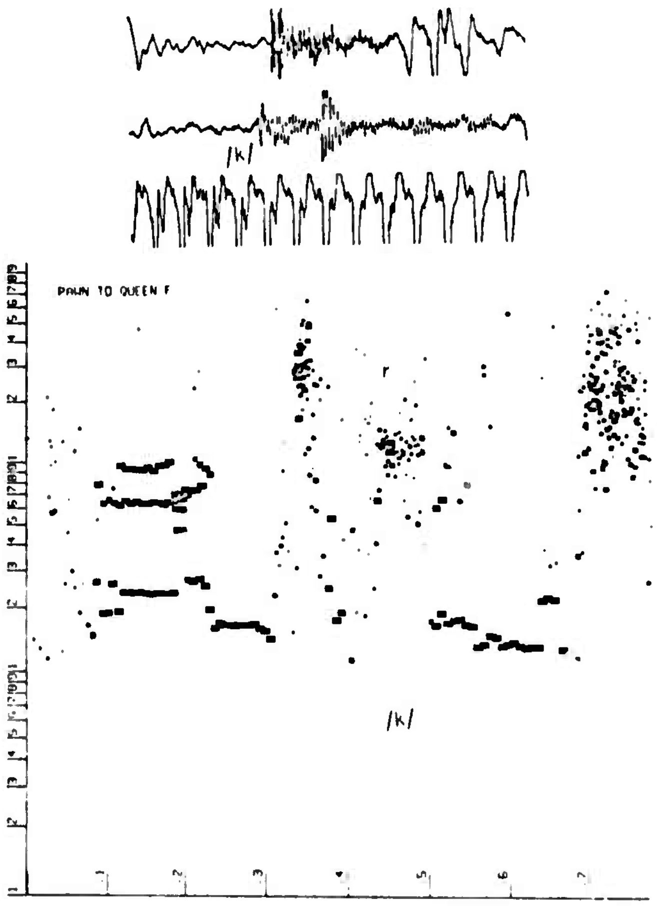
Given that neurons in the ear, as those in the other sensory modalities, often respond most vigorously to sharp discontinuities of the incoming signal, it is intriguing to speculate on the information provided by this common stop preview phenomenon and when it may be most useful. Its most obvious aspect is the cue it provides that a stop consonant follows. It is conceivable that especially in connected speech where stop consonants are often very brief, such redundancy of their presence may be helpful for stop consonant detection.

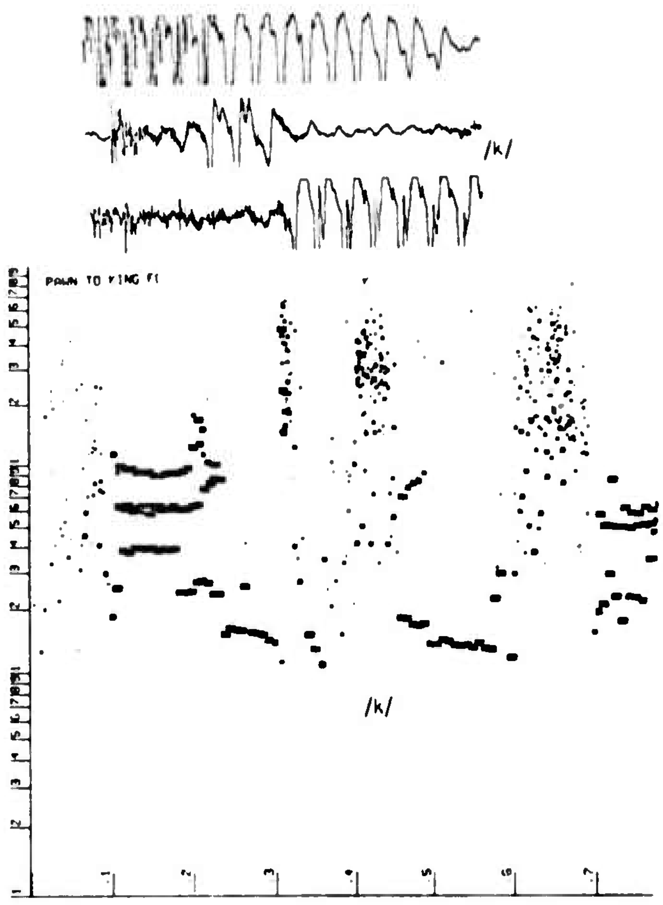
Allophones and Acoustic Correlates

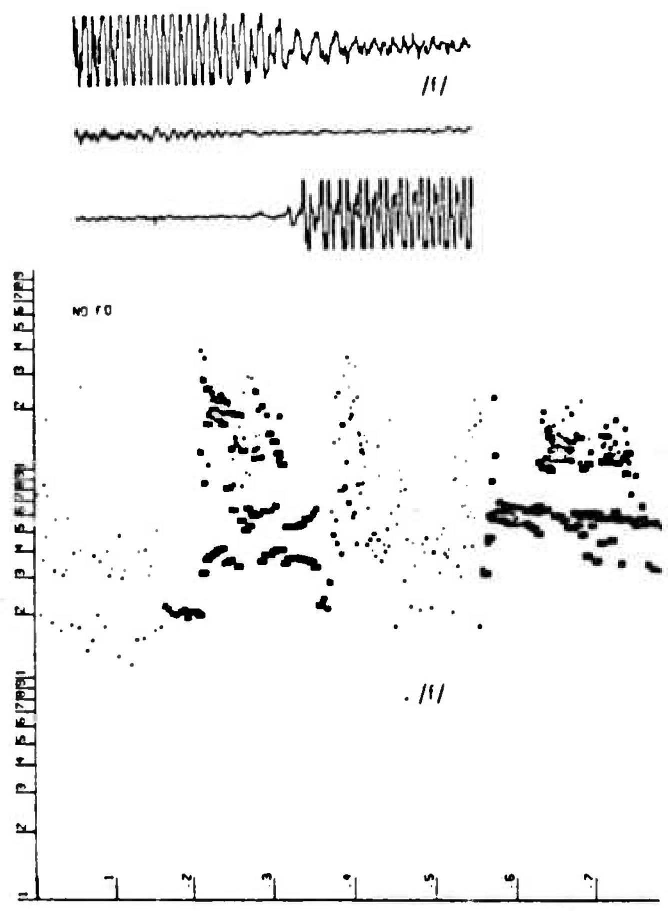
Using time-domain analysis, it is easy to compute, for example, characteristics of a /p/ release and compare these to those of a /t/ release. Certain general attributes become readily apparent. For example, the cycle-frequencies of the /p/ release are much more diffuse than those of the more concentrated /k/ release. A /t/ release, in comparison to both of these, usually has more energy concentrated at much higher cycle-frequencies. Given the same context, these attributes and other time-domain parameters are quite useful for consistently distinguishing between /p/, /t/, and /k/. However, the acoustic correlates of the release of a particular stop consonant in a given environment are often quite changed when this same phone occurs in a different context. Coarticulation effects thereby give rise to many allophones.

In the following examples are shown two allophones of the phone /k/, one rounded and one not. The utterances containing these are, respectively, 1)"pawn to queen four" and 2) "pawn to king four" (male speaker, JB). The release portion of the the rounded /k/ of "queen" is characterized by much lower cycle-frequencies than the release portion of the /k/ in "king". Rounding of the lips causes the vocal tract to be lengthened thereby lowering the cycle-frequencies emitted.









These examples demonstrate the importance of understanding coarticulation effects in the task of recognzing individual phones from acoustic information.

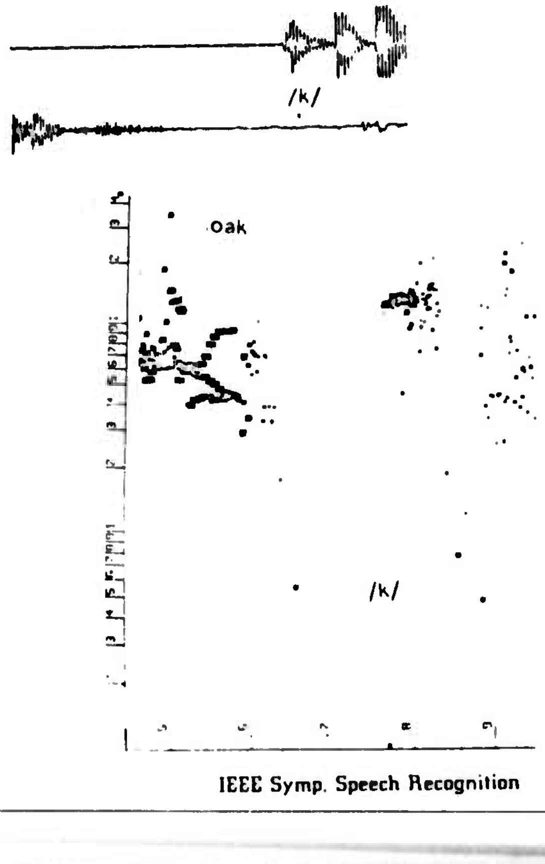
N.B. Readers interested in the specific acoustic correlates to the allophones of fricatives and stop consonants are referred to the author's Ph.D. thesis, 1974 (1).

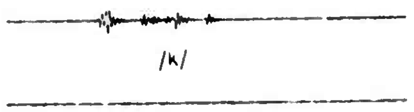
Acoustic-Phonological Phenomena

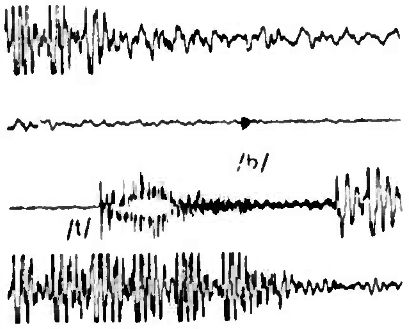
There are a variety of acoustic phonological phenomena which are commonly observed with time-domain analysis. Generally these phenomena are readily apparent in both the time-domain waveform and log inverse period plots. However especially when such acoustic events are either very low in amplitude or very brief in duration, or both, their existence is much more visually evident in in the log inverse period plots.

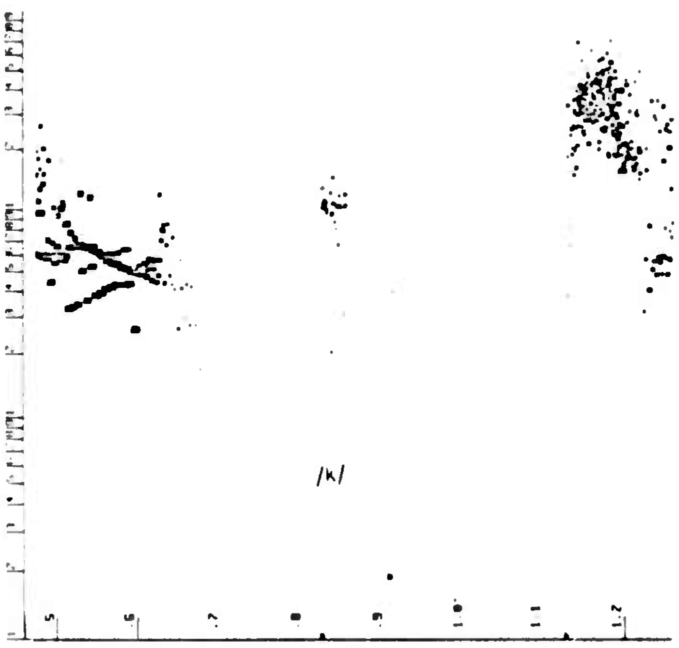
One very common phenomenon is the case where a fricative is characterized by a central region where the cycle-frequencies are lowered in relation to that phone's characteristic fricaticn frequencies. In the following example , the phone of interest is a rounded /f/ in the utterance "no foe" (female speaker, HN). The central mean frequency for the initial fricated region is 1079 Hz, for the central region is 693 Hz, and for the final fricated region is 1148 Hz. In addition, the first fricated region is much greater in amplitude than the central and final regions which are about equal in amplitude.

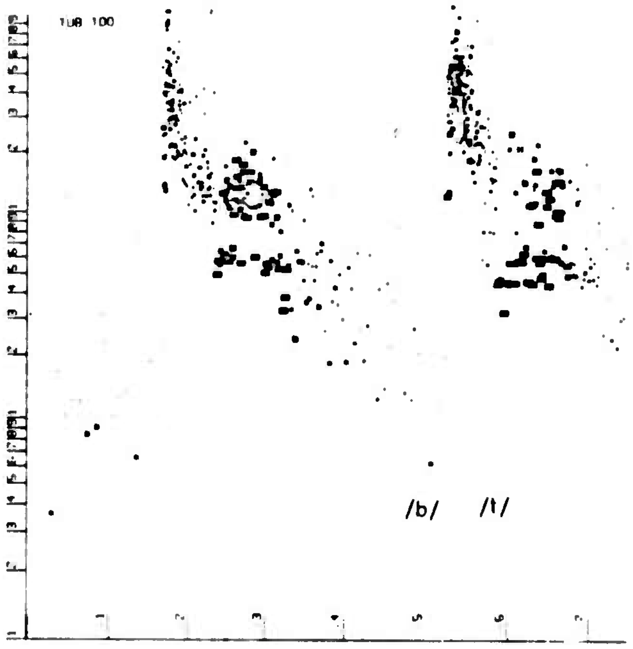
Another kind of event commonly occurs during the release portion of stop consonants. In the acoustic waveform, this portion is characterized by amplitude pulsing. The cycle-frequency composition of each of these pulses resembles that of the normal release portion of the same stop consonant when such amplitude pulsing is not present. Where aspiration occurs, it follows this amplitude pulsing, as it would a normal release. The following examples of both waveforms and LIP plots show such amplitude pulsed /k/s in the utterances 1) "soak me" and 2) "soak to" (male speaker, JA).





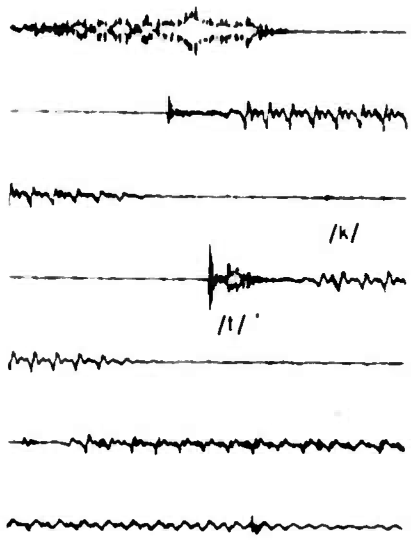


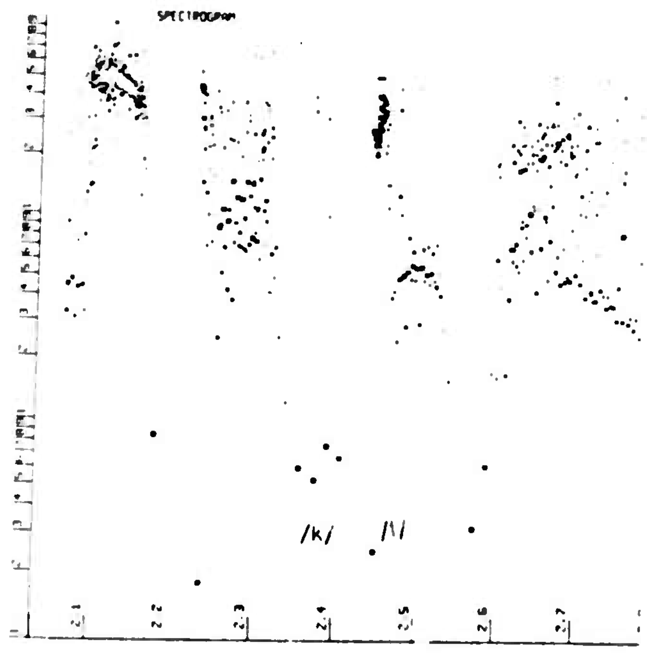




The next phenomenon regards the issue of the acoustic correlates of what are commonly referred to as "unreleased" stop consonants. Time-domain analysis reveals that often the stop consonants which are phonetically transcribed by linguists as "unreleased" or omitted, are acoustically characterized by the usual pause cycle(s), but with a very brief segment of high frequency energy which is analagous to a normal release segment, and which is sometimes followed by the transition cycle(s) leading into the next phone's acoustic events. This very brief segment consists of only a few cycles, often just one or two cycles where the the entire duration of this portion may be so short as to be less than 1 msec, and rarely more than 6 msec long. The temporal sequence of acoustic events characterizing these unreleased stop consonants is usually identical with that for released stop consonants except for durational aspects. The few cycles with high cycle-frequencies remaining in unreleased stop consonants are unreliable indicators for specific identification of the stop consonant. However, the information that a stop consonant has occurred and whether or not it was voiced does remain in most cases. The following example shows such an unreleased stop consonant, the /b/ in the utterance "tub look" (male speaker, EH). In this particular example, the segment with high cycle-frequencies is relatively long in duration, 4.2 msec, and is composed of 7 cycles preceded by normal voicing and pause cycles.

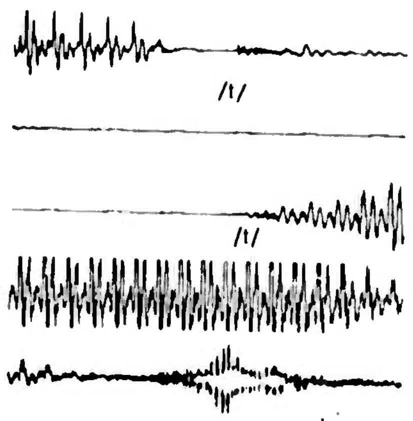
Another example follows where the same kind of acoustic event occurs during the course of 4 cycles lasting a total of 2.3 msec. It occurs for the /k/ immediately preceding the /t/ in the word "spectrogram" (female speaker, SM). Such short duration acoustic events are quite common in connected speech.

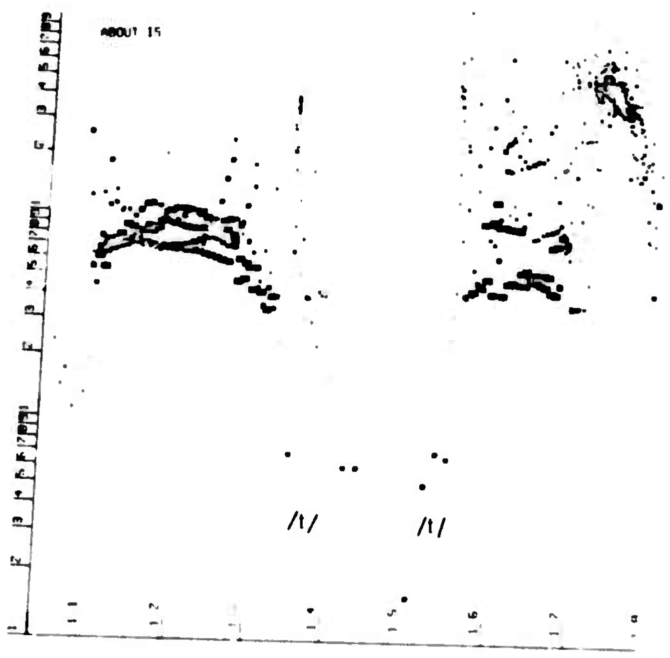




Frequently, as in both of these instances, an unreleased stop consonant is followed by another stop consonant which is released. Acoustic observations of a very brief stop consonant, often indicate that another stop consonant will immediately follow.

Another very common phonological phenomenon relates to the insertion of an extra stop consonant. This occurs when a syllable ends with a stop consonant and the next syllable begins with a vowel (or sometimes a liquid), even when there is a word boundary separating the two syllables. The speaker often articulate a normal stop consonant at the end of the first syllable as expected, but then repeats this same stop consonant when he begins the next syllable. When this happens it is not obvious to a human listener that a second stop consonant has been inserted by the speaker. In the following example of the utterance "about Israel" (male speaker, JB), the /t/ in "about" is repeated, even after a long interword pause of .17 sec, at the beginning of the initial vowel in the word "Israel". Acoustically both /t/s are complete in all respects.





Acknowledgements

The author is indebted to Jerome Y. Lettvin (Massachusetts Institute of Technology) for his helpful suggestions and criticism at the genesis of this research and to Raj Reddy (Carnegie-Mellon University) for providing the enthusiasm and facilities essential for implementing a thorough research investigation.

Bibliography

[1] Baker, J.M., A new time-domain analysis of human speech and other complex waveforms, Ph.D. Thesis, Carnegie-Mellon University, 1974

[2] Chang, S.H., G.E. Pihl, and J. Wiren, The intervalgram as a visual representation of speech sounds, JASA 23 (no. 6): 675-679.

[3] Ito, M.R., Investigation of time domain measurements for analysis and machine recognition of speech, Ph.D. Thesis, University of British Columbia, 1971.

[4] Kiang, N.Y.S., Discharge Patterns of Single Fibers in the Cat's Auditory Nerve, MIT Research Monograph, No. 35, 1965.

[5] Licklider, J.C.R. and I. Pollack, Effects of differentiation, integration, and infinite peak clipping upon the intelligibility of speech, JASA 20: 42-51, 1948.

[6] Licklider, J.C.R., The intelligibility of amplitude-dichotomized time-quantized speech waves, JASA 22: 820-823, 1950.

[7] Morris, L.R., The role of zero crossings in speech recognition and processing, Ph.D. Thesis, University of London, 1970.

[8]Rose, J.E., J. Brugge, D. Anderson, and J. Hind, Phase-locked responses to low-frequency tones in single auditory nerve fibers of the squirrel monkey, J. Neurophsiol. 30 (no. 4):767-793, 1967.

[9] Sakai, T. and S. Inoue, An analyzing equipment for the zero crossing interval and its applications to speech analyses, J. Inst. Elect. Commun. Engrs. Japan 39: 404-409, 1956 (in Japanese): English abstraction in Phys. Abst. 60: 110-1157, 1957.

[10]Shoup, J., The phonemic interpretation of acoustic phonetic data, Ph.D., University of Michigan, 1964.

SUB-LEXICAL LEVELS IN THE HEARSAY II SPEECH UNDERSTANDING SYSTEM

Linda Shockey and Lee D. Erman

Computer Science Department*
Carnegie-Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

The HEARSAY II system provides a uniform multi-level structure for representing the partial analysis of the utterance as it is being recognized and a convenient modular structure for incorporating new knowledge (i.e., processing capabilities) into the system at any level. This paper describes the sub-lexical levels chosen for the initial configuration (parametric, segmental, phonetic, surface-phonemic, syllabic) of the system and the kind of processing that is accomplished at those levels. The choice of levels is related to traditional phonological theories.

INTRODUCTION

The HEARSAY II (HSII) speech understanding system (whose system organization is described more completely in Lesser, et al., 1974) provides a unified structure for describing an utterance as it is being analyzed. This structure may be thought of as 3-dimensional, with the dimensions being level of representation (e.g., acoustic, phonetic, lexical, syntactic), time, and alternative possibilities. This structure is held as a single data base which the system maintains. HSII also provides a means for introducing **knowledge sources** (realized as computer programs) to work towards recognition; the knowledge sources (KS's) cooperate by examining and modifying this global data structure in a generalized form of hypothesize-and-test.

Earlier speech recognition systems have suffered from problems with internal levels of representation: in general, they have no clear distinction among such concepts as "acoustic", "phonetic", "phonological", "phonemic", etc.* The major difficulties caused by this fuzziness of representation are the inability to decompose the system so as to allow useful performance analysis of the various sources of knowledge and the inability to make use of results obtained by linguists and phonologists working along traditional lines. The HSII system, on the other hand, does not pre-specify the set of levels used in the data structure nor the set of knowledge sources; a particular system configuration is generated by defining the levels to be used and creating the knowledge sources to operate over them. Because the levels of representation are uniform and must be explicitly defined well enough for the KS's to interact through them in an independent manner, there is much more need and motivation to choose and delineate them in a less **ad hoc** manner than in previous systems.

This paper 1) describes the choice of the sub-lexical levels in the initial configuration (called **HSII-C0**) which is being implemented as the first test of HSII, 2) gives some feel for the kinds of processing occurring at and between those levels, and 3) relates those levels to traditional phonological theory.

* Two prime, but by no means exclusive, examples of this problem are the direct ancestors of HEARSAY II: the Vicens-Reddy system (Vicens, 1959) and HEARSAY I (Reddy, et. al., 1973a, 1973b).

THE LEVELS

The HSII-C0 configuration has five levels "below" the lexical level:

(6. Lexical)
5. Syllabic
4. Surface-Phonemic
3. Phonetic
2. Segmental
1. Parametric

At each level, a (potentially complete) representation of the utterance is formed, composed of units appropriate to the level.

At the _parametric level_, the speech is represented by vectors of parameters (e.g., spectral parameters), typically sampled, for example, every ten milliseconds.

At the _segmental level_, the utterance is described as being composed of labeled acoustic segments. Each segment represents an acoustically homogeneous section of speech (or a transitional segment) and is labeled in a way that describes its acoustic characteristics.

At the _phonetic level_, the utterance is represented by a phonetic description. This is a broad phonetic description in the sense that some acoustically dissimilar elements are grouped into perceptual units (e.g., silence + burst + aspiration may be represented by a single plosive symbol); it is a fine phonetic description in the sense that it is possible to specify articulatory modifications (retroflexion, nasalization) and degree of stress.

The _surface-phonemic level_ represents the utterance in units which can be thought of as phoneme-sized, with the addition of modifiers such as stress and boundary (word, morpheme, syllable) markings.

The _syllabic level_ represents an utterance as being composed of syllables.

At each level, there is an identical connection structure which allows the representation of sequences and (competing) alternatives. In addition, structural connections are also made across levels, relating how the elements at one level serve to support hypotheses at other levels.

* This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

PROCESSING

A knowledge source operates by reacting to a (sub-) structure built in the global data base by another KS; it adds new elements at some level or adds new connections between existing units. This operation of a knowledge source is triggered directly by the change to the structure, not by the other KS. Thus, a KS is not aware of other knowledge sources, but rather specifies the kinds of sub-structure and changes to which it desires to react.

At the sub-lexical levels, the general paradigm can be thought of as a rewriting scheme: a KS notices some structure and rewrites it as a different structure. In addition, it links the initial structure to its newly created one. Finally, if the new elements it is attempting to construct already exist (either previously created by itself or some other KS), then the structure is not duplicated; rather, new connections are made to the pre-existing structure.

For simplicity of exposition, the following description of these levels and processes assumes a bottom-up approach and linkages only between adjacent levels, but we will see below that these limitations are not in the system.

From the parametric level to the segmental level, the main action is to group acoustically similar samples and then label the segments. The segmentation scheme currently used in HSII-CO (Goldberg, et al., 1974) is parameter independent. At present, the parametric values for the segment target labels are determined from a corpus of continuous speech by one male talker, whic has been hand segmented and labeled with a fairly narrow phonetic transcription (using on the order of 75 labels). Each segment receives up to five different labels, each with a confidence rating.

Although the segment labels used are often also phonetic symbols, the level is not intended to be phonetic -- the segmentation and labeling reflect acoustic characteristics and do not, for example, attempt to compensate for the context of the segments or attempt to combine acoustically dissimilar segments into (phonetic) units. It is clearly necessary to improve on the method of target selection to accomodate speaker variation. Obviously, these targets can be established for any language, although we have dealt exclusively with English.

The segment labels are actually defined through a set of features: each segment is defined as having a ternary value (+, -, or 0). Other than being ternary, as opposed to binary, these features bear some resemblance to the the Jakobson-Fant-Halle (195) feature set. The use of features creates an indirectness of reference which isolates the processing algorithms (knowledge sources) from any particularly chosen set of segment labels; thus different parametric representations may use different sets of labels (i.e., they need only be defined in terms of the feature vectors). This feature representation is also a means of creating an algebra for manipulating the segment labels: for example, the five alternative labels assigned to each segment may be combined by combining their feature vector definitions (assuming a value of -1 for -, +1 for +, and 0 for 0 and using the confidence measures as weightings). The values of individual features of such a combined vector may be used directly (e.g., to determine if a segment is "voiced" or "nasalized") or the entire vector may be used to derive a new label, which will tend to be an "average" over the input labels.

Going to the phonetic level, the main activity is hypothesizing phones from the labeled acoustic segments, using the adjacent acoustic segments and previously recognized phones as context. This hypothesization may take several forms:

1) A single segment may be propagated as a single phone, with, perhaps, some relabeling. In this case the phone is one-to-one with its acoustic segment. Moreover, patterned errors caused by allophonic overlap are dealt with here. For example, a rule at his level could say, "nasalized [OW] might be velarized [AX] if it is found before [L]."

2) One phone may be synthesized from several similar adjacent segments. This form of combining can be thought of as a way of correcting errors of the segmenter that require contextual information.

3) A phone may be synthesized from several similar adjacent segments. For example, a stop may be generated from a silence followed by a segment of noise.

4) A phone may be generated from within one or across two (or more) segments. For example, the sequence of [IYn T]* may become [IY N T] at the phonetic level, expressing the idea that the phone N may be acoustically detectable only as a nasalization of the preceeding vowel. However, [IYn N] would be rewritten as just [IY N], since the nasalization is predictable from the environment.

5) Phones may generated using combinations of the above.

The broad phonetic transcription at the surface-phonemic level is linked to the dictionary pronunciations (from the the lexcical level). This association process uses phonological rules which rewrite symbols at the surface-phonemic level. For every phonetic element assumed to be present, a determination is made as to what underlying (phonemic) element or sequence of elements could have generated it. For example, an utterance-final [N] could have been derived from any of [N], [NX], [N T], or [N D]. Each possibility which is generated is given a confidence rating depending upon how strong the initial identification is and upon what support is derived from environmental evidence. Matches are then made of temporal and segmental properties between surface-phonemic and lexical items. Of course, at this point there is strong interaction with the syntactic and semantic components of the system working from the higher levels.

The syllabic level is one which receives only cursory attention in the present implementation of the system. We hope to use it in the future as a repository of prosodic information. Also, this looks like a very promising level for doing effective lexical retrieval in terms of the size of the syllable unit in relation to the size of words.

It should be understood that although this structure has been presented as a strict sequence of bottom-up processing through adjacent levels, there are no such restrictions in the system; in fact, much of the action comes from top-down processing and level skipping. For example, if a word is hypothesized at the lexical level which has elements different than those generated from below, it is possible to probe down through the levels, hunting harder for evidence that substantiates the word hypothesis. As an example of level-skipping, given that an hypothesizer at the syntactic or pragmatic level has suggested that a sentence may be a 'yes-no question', an immediate skip may be made to the parametric level to investigate pitch contours. These various kinds of actions of top-down, bottom-up, and level-skipping can all be happening simultaneously, as the knowledge sources are executed asynchronously and in parallel.

* We use the notation [IYn] here to mean "nasalized [IY]".

LEVELS AND PHONOLOGICAL THEORY

In the past several decades, two major phonological theories have achieved prominence. The **phonemic theory** (Gleason, 1966, Hockett, 1955, Harris, 1951) is based on the tenet that there are discrete levels of analysis on the morphophonemic, phonemic, allophonic, and phonetic levels and that each of these levels can be mapped onto its neighboring levels by the use of a set of distributional statements plus a statement for each segment regarding its free variation properties. In this theory, the phone is the surface-level entity: that which is actually articulated. The other levels are perceptual or statistical constructs.

This theory has been attractive to builders of speech recognition systems for two unrelated reasons: 1) its separate levels are relatively easy to deal with in a computer system and 2) several influential people in speech recognition have been trained in and/or have contributed substantially to phonemic theory.

The second general class of theory, which has enjoyed popularity more recently, is called **generative phonology** (Chomsky-Halle, 1968, Postal, 1968). In general, it assumes only two fixed levels: some sort of underlying representation in abstract for possible sequences of sounds in a given language (the nature of which is much debated), and the phonetic output level (or something very close to it). Connecting these two levels is a set of phonological rules, frequently thought to be ordered, which change properties of segments and possibly add or delete segments. These rules can be optional or obligatory. They can be compared to a series of filters: given a sequence of elements destined to be articulated (including all types of boundaries), the entire string is fed into the first filter. If it is able to modify an element or group of elements in the input string, it does, otherwise it lets the string pass unchanged. Of course, if the filter is an optional one it may or may not be switched in. Then the string is passed to the next filter. In general, alternations between possible surface pronunciations of a given base form are caused by an optional rule having applied or not applied. The major point here is that there are very many output levels for these filters, most of which can be inputs to others.

At present, researchers trained in each of these theories are occupied in automatic speech recognition; some are trained in both. It seems that a synthesis of the theories, or at least an agreement as to terminology, would be desirable, since workers in ASR quite frequently use the idea of distinct levels of analysis (phonetic, allophonic, phonemic, etc.) but are also interested in using phonological rules in a generative rather than a descriptive sense. Perhaps attempts at building systems, such as HEARSAY II, which explicitly span the full range of levels and make efforts at conceptual cleanliness will prove an incentive and test-bed for such a synthesis.

Due partially to this mixed theoretical framework, we experience difficulty in finding reasonable terminology for at least one of our levels. The parametric and segmental levels seem to be largely extra-theoretical, having more to do with theories of speech perception than directly with phonological theory. The term 'phonetic level' seems well-motivated in that this level attempts to postulate a phonetic transcription of the input or to generate one from a hypothesized word. The syllabic level is probably more related to acoustic-phonetic studies, though some phonologists use the syllable boundary in rule writing. But the level we call 'surface-phonemic' is not easily characterized in terms of either of the theories mentioned above in most cases. The hypotheses generated from below (typically the phonetic level) represent a proposed phonemic transcription of just those elements which are identifiable from the speech input; the hypotheses genereated from above (e.g., from lexical or syntactic knowledge) include most of the possible alternative sequences of allophonic tokens which can be related to the dictionary spelling, but represented very broadly. This puts the surface-phonemic level on a theoretically non-existent level somewhere between allophonic and phonemic. In generative terms, the 'surface-phonemic' level is more underlying than the output of the Chomsky-Halle (1968) phonological component since it is a very broad transcription. It is a form intermediate between underlying and surface forms; but it is a level which we find useful despite its lack of theoretical ancestry.

ACKNOWLEDGMENTS

We wish to thank Raj Reddy, who has inspired much of the formulation of these ideas, Victor Lesser and Richard Fennell for their efforts in the design of HEARSAY II, which have spurred much of our efforts, and Gary Goodman, who has been very constructively reactive to our ideas.

BIBLIOGRAPHY

Chomsky, N. and M. Halle (1968) **The Sound Pattern of English**, Harper and Row, N.Y.

Gleason, H. A. (1961), **An Introduction to Descriptive Linguistics**, Holt, Rinehart, and Winston, N.Y.

Goldberg, H. G., D. R. Reddy, and R. L. Suslick (1974), "Parameter-Independent Machine Segmentation and Labeling," Proc. IEEE Symp. Speech Recognition, Pittsburgh, Pa., pp. 106-111, (this volume).

Harris, Z. (1951), **Structural Linguistics**, University of Chicago Press.

Hockett, C. F. (1955), **A Manual of Phonology**, IJAL Memoir 11, Waverly Press, Baltimore.

Jakobson, R., G. Fant, and M. Halle (1951), **Preliminaries to Speech Analysis**, MIT.

Postal, P. (1968a), **Aspects of Phonological Theory**, Harper & Row.

Reddy, D. Raj, Lee D. Erman, and Richard B. Neely (1973a), "A Model and a System for Machine Recognition of Speech," **IEEE Trans. Audio and Electroacoustics**, AU-21, 3, June, 1973, pp. 229-238.

Reddy, D. R., L. D. Erman, R. D. Fennell, and R. B. Neely (1973b), "The HEARSAY Speech Understanding System: An Example of the Recognition Process," Proc. 3rd Inter. Joint Conf. on Artificial Intel., Stanford, Ca., pp. 185-193.

Vicens, P. (1969), "Aspects of Speech Recognition by Computer," CS 127, AI-85, Comp. Sci. Dept., Stanford Univ. (Ph.D. thesis), Stanford, Ca.

INFERENCE AND USE OF SIMPLE PREDICTIVE GRAMMARS

Elaine Rich
Carnegie -Mellon University*
Pittsburgh, Pa. 15213

One use of syntactic knowledge in a speech understanding system is to focus the system on the most probable paths as it is attempting to understand an utterance. This function is frequently performed by a parser similar or identical to the one used to generate a parse of the entire utterance. However, it is possible to perform a large part of this function without incurring the overhead of generating many partial parses, most of which will eventually be thrown away. This is done by using a simple probabilistic grammar which, given a string of already recognized words, can predict the words which can precede or follow the string, and associate with each such word the probability that it will occur. The system can then consider the most likely possibilities first. If they are rejected by the lower level knowledge sources, then the less likely possibilities can be considered.

A knowledge source for the Hearsay II system (Lesser,1974) has been constructed to do this. The data used by this syntactic knowledge source consist primarily of a collection of sentence fragments of varying lengths, each of which has associated with it a list of words which can precede it and a list of words which can follow it, along with the probability that each of those words will occur in that environment. These fragments may contain both specific words and grammatical classes. The fragments are arranged by the word immediately adjacent to the word to be hypothesized. The program uses a lexicon and a grammar which provide it with the information it needs. The lexicon contains an entry for each word in the vocabulary which specifies the grammatical category to which the word belongs. The grammar specifies, for each grammatical category, the fragments which begin and end with that category and the words which can adjoin them (and the probability associated with each word). To predict words at a given point in the utterance, the knowledge source looks up the word of the partially recognized utterance which is adjacent to the word to be predicted. Listed for the part of speech to which it belongs will be strings of arbitrary length starting with that word (for predicting to the left) and ending with that word (for predicting to the right). The program uses the longest such string which matches the utterance fragment and predicts the alternatives listed as occurring on the desired side of the fragment.

Since storing long strings to be used for prediction incurs a great deal of overhead, both in terms of space and in terms of the time required to check for a match between the stored strings and a recognized piece of the utterance, it is desirable to store long strings only if the use of additional words causes a significant increase in the accuracy of prediction. Experiments will be conducted to discover when increasing the length of the strings ceases to cause such an increase in performance.

* This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

The criteria for assigning words to grammatical classes in this system are well defined and are not necessarily the same as in the traditional grammatical system with nouns and verbs. The first criterion is to maximize the amount of information known about the environment of a word, given its grammatical class. Thus, words which tend to occur in the same environment should be in the same class. The second criterion is the restriction of the number of classes in order to cut down on the number of sentence fragments to be stored as well as the number of possible alternatives adjoining each of those strings. A program is being developed which will read a corpus of utterances and construct grammatical categories from the words of the corpus using the maximization of information criterion. As with the question of how many words should be used to define the environment, the question of how many grammatical classes to use will be answered empirically by observing the point at which the addition of more classes does not significantly improve the predictive ability of the knowledge source. The principal problem in getting the categorization program to do very well is the need for a corpus large enough so that each word occurs enough times to be able to know what environments it can occur in.

The program which constructs grammatical classes can also construct, from the corpus, the lexicon and grammar needed by the knowledge source. Thus it should eventually be possible to have the machine both construct the grammar as well as use it. One result of this is that it should be relatively easy to construct a grammar based on a new corpus, thereby allowing the sytem to recognize utterances pertaining to a new task.

Bibliography

Lesser, V.R., R.D. Fennell, L.D. Erman, and D.R. Reddy, "Organization of the Hearsay II Speech Understanding System", IEEE Symp. Speech Rec., April, 1974 (this volume).

REAL-TIME LINEAR-PREDICTIVE CODING OF SPEECH ON THE SPS-41 MICROPROGRAMMED TRIPLE-PROCESSOR SYSTEM

Michael J. Knudsen
Carnegie-Mellon University
Pittsburgh, Pennsylvania

Summary

Markel's autocorrelation method for linear predictive coding of speech [1] has been implemented on the SPS-41, a commercially available system composed of three dissimilar microprocessors working in parallel. Using user-written microcode, one processor performs I/O and master control, the second handles loop indexing and counting, and the third does the actual arithmetic on data. Such parallelism allows 2M I/O operations and 4M multiplications per second, but actually realizing this potential requires fresh approaches to some old algorithms, e.g., a new autocorrelation scheme with several valuable properties. Inverting the autocorrelation matrix in 16 bits of fixed point also poses problems. The present program converts 256 words of 13-bit samples into 14 coefficients at 100 frames per second.

Review of Markel's Method

Motivation

Our major interest in Markel's method at C-MU is to find the resonance spectrum of the vocal tract for each frame of speech, where a frame is about 200-300 samples for a 10 kHz sample rate. The next step after this (not covered here) is to identify the formats or otherwise compare the resonance curve with a standard set of corresponding data for various phonemes, in order to identify the phoneme spoken.

A straightforward high-resolution spectrum of the frame (as by an FFT) will not do, as it will have not only the frequency response of the vocal tract, but will also superimpose the spectrum of the excitation source. This will either be a dense series of sharp peaks and valleys from the glottal pulses in voiced speech, or a random jagged curve from the white noise in unvoiced speech. In either case the many extraneous peaks and valleys mask out the desired formant peaks.

Overview

Markel's method is a form of deconvolution, or separating the effect of the driving function (unwanted in our case) from that of the driven system (the desired vocal tract response). Thus the smooth resonance spectrum of the vocal tract can be obtained. (The excitation signal can also be identified and used for pitch extraction.)

Markel derives an inverse filter for each frame of speech signal. Such a filter attempts to destroy the signal input, i.e., reduce it to minimum energy and information content, either white noise or zero. The frequency response of this filter must be the inverse of the spectrum of the signal for which it was designed. However, by judicious selection of its length, the filter can be made capable of wiping out the gross frequency characteristics of the signal (which correspond to the formant resonances), but unable to follow the fine detail of the input spectrum (due to the excitation source). Thus the filter's frequency response is inverse to the desired vocal tract response, but not to the undesired excitation. Since we generally work with logarithmic (dB) frequency response scales, and log(1/x) = -log(x), we need only reverse the sign of the inverse filter response (as by viewing it upside down) to plot the frequency response of the vocal tract.

Nature of Inverse Filter. The filter is finite-impulse-response, all-zeroes, and implemented in direct feed-forward form with unit delays. Such a filter of length M is represented as:

$$A(z) = 1+\sum_{i=1}^{M} a[i] * z\uparrow(-i)$$

Markel's algorithm designs the filter for each frame by specifying the M values a[1], a[2], . . . , a[M].

The frequency response of any filter is defined as the spectrum of the output resulting from a single unit impulse input. However, the filter defined above will respond to a unit impulse simply by outputting a 1 and then reading out its coefficients in order, followed by zeroes forever. Therefore a discrete Fourier transform (DFT) applied directly to the series

1, a[1], . . . , a[M], 0, 0, . . . , 0, . . .

followed by magnitude, logarithm, and negation will compute the vocal tract resonance spectrum.

The Algorithm

Autocorrelation. Given a frame of L digitized speech samples, x[1] thru x[L], the first step in deriving an inverse filter of M stages is to compute the autocorrelation vector R = r[0], r[1], . . . r[M], where

$$r[n] = \sum_{i=1}^{L-n} x[i]*x[i+n]$$

Markel claims [1] that much better results are obtained when the input is multiplied by a non-rectangular window. Since our own tests have not refuted this, and our autocorrelation method permits windowing at low overhead, we precede the autocorrelation by a Hamming windowing:

$$x[i] := 0.5*(1-\mathrm{COS}(2*\mathrm{PI}*i/L))*x[i], \qquad 1\le i\le L$$

Matrix Inversion. The filter coefficients a[i] are obtained by solving the system of linear equations R*A=B, where

A transposed = [a[1] a[2] . . . a[M]]

B transposed = [r[1] r[2] . . . r[M]]

and

$$\underline{R} = \begin{bmatrix} r[0] & r[1] & r[2] & \ldots & r[M-1] \\ r[1] & r[0] & r[1] & \ldots & r[M-2] \\ r[2] & r[1] & r[0] & \ldots & r[M-3] \\ r[3] & r[2] & r[1] & \ldots & r[m-4] \\ \cdot & & & & \\ \cdot & & & & \\ rM-1 & rM-2 & rM-3 & \ldots & r[0] \end{bmatrix}$$

Since the "autocorrelation matrix" R is symmetric, positive definite, and of Toeplitz form, it can be solved in k*M↑2 steps rather than the generally needed k*M↑3, k a constant. Our method is described under Implementation.

Spectrum. An N-point real DFT is applied to 1, a[1], . . . , a[M] followed by N-M-1 zeroes. Since the a[i] are real, the magnitude of the transform is symmetric about its center, and only the first N/2 frequency bins need be computed.

Optimum Values. Markel's experiments show [1] that for a sampling rate of 10 KHz, best results are obtained with M=14 and 200<L<300. We use M=14 and L=256. Our DFT has N=256 input points and thus outputs 128 bins for a frequency resolution of about 40 Hz.

"Real-time" Defined. In this paper we define "real time capability" as maintaining a frame rate of 100/sec or better.

Structure of the SPS-41

The SPS-41 is built by Signal Processing Systems, Inc. of Waltham, Mass., costs about $30,500, and occupies the same amount of rack space as a PDP-11/20 minicomputer.

Design Philosophy

The SPS-41 achieves high speed with modest hardware by decomposing algorithms according to the sometimes-overlooked fact that even a numerical analysis procedure spends only about 25% of its time computing on the data, with the rest divided about equally between loop administration and memory stores and fetches. Any concurrent I/O operations will further reduce the fraction of time devoted to actual data processing.

Assigning parts of a computational task to multiple processors according to their nature (calculation, loop indexing, or store/fetch), gives a form of parallelism quite distinct from either the ILLIAC-IV approach or a pipelining of minis where each does one phase of the Markel analysis. The first form is expensive and possibly not suitable for linear prediction computation. The second requires each mini to have powerful arithmetic units (hardware multiply), but since each machine implements all aspects of one phase, its multiplier is idle 3/4 of the time.

Putting one aspect of all phases of the total process on each processor, as in the SPS-41, permits restricting expensive multipliers to the one section that needs them, and conversely having no loop-testing facilities in the arithmetic section. This is just one example of the cost-saving specialization of processors achievable by this form of algorithm decomposition.

Individual Processor Characteristics

General. All sections deal with 16-bit 2's complement data and have a 200-nsec instruction cycle time.

Arithmetic Section (AS). The AS contains three data memories, a read-only sine/cosine table, four multipliers, six summers (adders), and a 16x64 microcode store. The basic data type is a complex word consisting of real and imaginary halves, each 16 bits. However, the AS allows the two halves to be treated separately as reals.

For a complex multiply, each of the four multipliers generates one of the terms, and two summers built into the multiply section form the real and imaginary outputs. Either the high or low 16 bits of the 32-bit product may be taken, but not both. The high/low choice must be made when the multiply is done, not afterwards. Getting both halves of a result for double precision requires repeating the multiply with the same inputs. Products may be scaled up or down a maximum of two bits; if the result would overflow, it saturates to +-2↑15; saturation cannot be disabled. Other modes besides complex may be requested, e.g., conjugate, matrix, and twin real.

There are also two complex summers (four real adders) under direct microcode control, plus one complex accumulator whose outputs can be scaled like the products.

To reduce the tendency for processing to be I/O-bound, the AS has data memories to enable buffering, large-radix FFTs, etc. AS data storage consists of two identical memories HI and LO each with 64 complex words, and the COEFF memory with 32. These may all be read and written during one AS instruction. The only ROM in the entire SPS-41 is TRIG, whose sines and cosines are used for Hamming windows and Fourier transforms.

The AS logic is 4-bit byte serial and requires 5 clocks or one usec to complete an instruction. Thus 4M real multiplies and 6M adds per second can be achieved.

The AS is a passive slave without even a program counter. The microinstruction for each cycle is selected by the Index Section, as are the scale factors and read/write memory addresses.

Index Section (IS). The IS is the controller for the AS. Rarely does any data (meaning speech-related data) pass through it. The instruction set is oriented toward the byte shifts, bit extraction, and rapid condition testing required for loop indexing and control. It has a 32x16 general memory plus 32 16-bit registers including 7 accumulators, 4 control interfaces to the AS, and 15 trap registers.

The IS is a true computer with a program counter. There are only 48 words of 32-bit program store, but these are augmented by the Trap system, the most interesting feature of the IS. No direct test, branch, or halt instructions exist. Instead, 4 bits of every program reference one of the 15 trap registers. (Trap 0 does not exist and signifies "no test.") Each user-loaded trap register can hold 16 bits worth of tests and branch address for the price of just 4 bits in the instruction. Furthermore, any instruction can test and branch or halt on its results for free. Thus both length and breadth of program store are conserved.

Input-Output Processor (IOP). The IOP interfaces the SPS-41 with the outside world and coordinates the 3 sections of the 41. It is a universal device controller, where "device" includes the rest of the SPS-41, anything attached directly to the 41's I/O bus, or anything on the PDP-11's Unibus. The IOP can halt the IS-AS pair, and later continue them or re-initialize them and restart the IS at any point in IS program store.

Up to 16 programs or "channels" can be timeshared by the IOP on a fixed priority basis. At each 200-nsec clock, the cycle goes to the highest-priority channel which is not waiting for an unfulfilled external status condition, e.g., core memory fetch complete. (Note that PDP-11 memory is treated as a peripheral device, as suits it's relative slowness.) Since there are 16 copies of the program counter and the accumulator files, the equivalent of an interrupt is serviced within 200 nsec with zero overhead.

The IOP has a 265x23 program store and a 256x16 data memory, plus external registers for peripherals (including PDP-11 core) and four bidirectional data interfaces to the AS. Each instruction can operate on two separate operands and put the result in a third location, making the IOP a 3-address machine. Separate instructions must be executed for tests and gotos or subroutine calls, unlike the IS.

SPS-41 Implementation of Markel's LPC

The System

The SPS-41 is interfaced as a peripheral to a PDP-11/20 (now 11/40) mini. Other relevant peripherals are two magnetic tape drives and a pair of author- constructed 12-bit digital to analog converters (DACs). Currently the digitized speech data must be imported from our PDP-10 via magnetic tape. The PDP-11 reads successive frames of raw data from this tape into core; the SPS-41 performs the Markel analysis on each frame and writes its resonance spectrum into another core buffer; and the PDP-11 writes this onto the output tape mounted on the second tape drive. Each input frame and its resonance spectrum are also displayed on an oscilloscope connected to the DACs; this immediate viewing is helpful in evaluating the quality of the SPS-41's computation. The output tape is then transferred to the PDP-10 for more extensive reviewing. While this tape to tape operation is hardly "real time," the system is capable of 100 frames/sec and will run in real-time mode once the required analog to digital converter and clock have been installed on the PDP-11 Unibus. Quality and accuracy are still the major goals, as we are not yet fully satisfied.

Implementation of 3 Phases

The SPS-41 analyzes the signal in three phases: Hamming window and autocorrelation; matrix solution for the filter coefficients a[i]; and the log magnitude DFT of the a[i]. Since the programs can not all fit in the 41 (especially the IS), a small swapper program residing in the IOP is called at the conclusion of each phase to roll in the programs, constants, and data initializations for the next phase. Since the IOP can access any memory in the 41, it can load these from special core images called <u>overlays</u>. The swapper is loaded by the PDP-11 while the 41 is in external (passive slave) mode.

<u>Autocorrelation.</u> The usual scheme for the short term autocorrelation <u>R</u> of an input <u>X</u> of length L up to the Mth lag is:

```
real array r[0:M], x[1:L];    real sum;    integer n, i;
for n:=0 step 1 until L do begin "lag_sums"
   sum:=0.;
   for i:=1 step 1 until L-n do sum:=sum+x[i]*x[i+n];
   r[n]:=sum;
end "lag_sums";
```

(Note: "<u>Real</u>" and "<u>Integer</u>" are used here only to distinguish data and indices, respectively. "Real" values are of necessity fixed-point numbers in the SPS-41.)

This procedure reads most of the x[i] 2(M+1) times. The accessing pattern is M+1 sweeps thru the array <u>X</u>. In an ordinary computer this is no problem; but L=256 points will not fit conveniently in the 41's AS data memories, and larger numbers will not fit at all. Thus the 41 would make almost 2*(M+1)*L reads from PDP-11 core with the above procedure, even though there are only L unique values. Since core is to the 41 what drums are to a conventional computer, the memory-boundedness of the above scheme is intolerable.

Note that each x[i], M<i<L-M, is involved in 2M+1 products: M with x's of lower index , one with itself, and M with x's of higher index. Using just enough AS memory to hold an x[i] and the other x's involved with it, we can compute all of x[i]'s contributions to the lag sums while x[i] is in the AS memory; thus each x[i] need be fetched from core only once. In our case, M=14<<L=256, giving a definite reduction in AS memory needs.

We use the 3 AS data memories as follows:

	LO	COEFF	HI
Bottom	x[i]	x[i]	r[0]
	x[i+1]	unused	r[1]
	x[i+2]	unused	r[2]
	.	.	.
	.	.	.
Top	x[i+M]	unused	r[M]

To compute the contribution of the <u>pivot</u> value x[i] in COEFF to the partial sums r[0] thru r[M], multiply the pivot by the x-value in each row of the LO buffer, and add this product to the r-sum in the same row, i.e.,

$$r[k]:=r[k] + x[i]*x[i+k], \quad 0\le k\le M$$

Now shift the the LO buffer down one, discarding the x[i]. Copy the new buffer bottom x[i+1] into COEFF as the new pivot. Fetch the next x-value x[i+M+1] and put it at the top of the buffer. Then repeat the products and sums. Repeat the above until x[L] has been fetched into the buffer. Continue from there by fetching zeroes in place of the non-existent x[>L]'s, until x[L] has been the pivot. Then stop. (The procedure is initialized by filling the buffer with x[1] thru x[M+1] with x[1] as the pivot for the first set of products.)

Note that the products of x[i] with itself and x's of higher index are formed while x[i] is the pivot, and its products with lower indices occurred previously as x[i] worked its way down thru the buffer. Thus the autocorrelation can be computed using just 2M+3 words of memory (including the r[i]) and fetching each input x only once.

Also note that M+1 multiplications and additions take place between x-fetches, so the data rate of the input storage medium may be lower by a factor of at least 2M, compared to the conventional method. This is important in any system using two-level, cache, or virtual storage. The single fetching of the x[i] in increasing order not only assures optimum efficiency under paging systems, but also suggests a real-time autocorrelation scheme in which each data point's contribution to the running lag sums is computed as soon as it comes from the outside world. By doing r[k]:=C*r[k]+x[i]*x[i+k], where C is almost 1.0, earlier contributions to the running sums will exponentially decay and real-time displays of long-term continuous signals could be displayed.

Important here is the elimination of a core-to-core Hamming windowing. Since each x[i] is fetched but once, it is multiplied by the appropriate point on the Hamming weighting upon entry to the AS, before being placed on top of the buffer. Windowing contributes only 7% overhead to our SPS-41 autocorrelation. Our program does not shuffle the buffer for each new input, but instead the IS maintains pointers to the buffer's top and bottom, which crawl around the LO memory.

The 13-bit speech input values are regarded as ranging from -1.0 to almost +1.0, thus are scaled x2↑-13. The products are left at x2↑-11 scale, allowing for sums from -16.0 to almost +16.

<u>Matrix Solution.</u> Procedures for inverting this form of Toeplitz matrix date back to Levinson [2] and Robinson [3], were adapted by Markel [1], and later simplified by Markel and Gray [4] who eliminated 3 of the 7 steps as redundant. This redundancy was also discovered by the author. All versions of the algorithm are iterative, and after the nth iteration an nth-order inverse filter has been designed.

The algorithm implemented on the SPS-41 is:

```
real array R[0:M], comment input from autocorrelation;
  A[0:M], comment output filter coefficients;
  TA[1:M]; comment temporary storage;
real alpha, beta, C;    integer n, i;
comment Initialize;
a[0]:=1.0;  alpha:=R[0];
for i:=1 step 1 until M do begin
  A[i]:=0.;   TA[i]:=0.;
end;
;
for n:=1 step 1 until M do begin "Iterations"
  beta:=0;
  for i:=0 step 1 until n-1 do  beta:=beta+A[i]*R[n-i];
  C := -beta/alpha;
  for i:=1 step 1 until n do  TA[i]:=A[i] + C*TA[n-i];
  for i:=1 step 1 until n do  A[i]:=TA[i];
  alpha := alpha + C*beta;
end "Iterations";
```

All arithmetic is done by the AS except the divide, which is programmed in the IOP by the usual minicomputer techniques. For M=14, the entire procedure takes less than one msec, of which divisions account for half. Scaling is x2↑-13, allowing values from -4.0 to almost +4.

Log-Mag DFT. The AS computes the complex Fourier transforms one at a time for each frequency from 0 to 5000 Hz in steps of about 40 Hz. Using complex conjugate multiply, the AS then finds the squared magnitude and passes both halves of the 32-bit product to the IOP. The IOP locates the most significant bit and encodes its position as the 5-bit exponent of the base-2 log, and the top 7 bits of the normalized value are used to index a 128-word table of logs from 1.0 to almost 2.0 (currently kept in PDP-11 core) to fill in the 8-bit mantissa. After rounding the 13-bit log to 12 bits, the IOP writes it into the core output buffer and is ready for the next value from the AS.

Since the AS computes a 256-point DFT on only 15 nonzero points, and only the first half of the results are unique, and the IOP logarithm procedure can handle only one value at a time, it is just as practical for the AS to compute the Fourier transform by direct integration, rather than by any "fast" FFT techniques. This phase takes about 2 msec. While a pruned FFT could do its part faster, the log-mag part would be slowed down; thus we do not intend to switch to an FFT.

Results and Conclusions

Results. The system has been fully operational since January 1974. However, it tends to give obviously incorrect results on strong voiced segments (vowels). Sibilants never fail. Early problems with saturated products in computing Beta in matrix inversion and the Fourier sums were solved by scaling down an extra bit. Disabling the AS saturation logic, if possible, would also have solved these problems.

Conclusions. The remaining problems lie in the matrix inversion. Markel and Gray [4] show that if C≥1.0 at any iteration, the inversion has failed due to numerical errors. PDP-10 simulations of the fixed-point arithmetic reveal several excessive C-values during those speech frames for which faulty SPS-41 output is observed. Excessive C's usually result from relatively small values of Alpha and Beta; this suggests too much loss of significance in the x2↑-13 scaling. However, the current scale's range of -4. to +4. is often needed, so the conclusion seems to be that more bits/word are needed. Markel and Gray [4] state that 23 bits are required. Thus future efforts will probably be devoted to converting some of the operations to double precision; the question is, which operations can be safely left in single?

Note that taking the high half of a product amounts to truncation, without roundoff. We have altered the matrix inversion AS program to achieve rounding at no extra cost as follows: Appropriate constants are kept in the imaginary halves of all data words, such that their contribution to a complex conjugate multiply is one-half the value of the LSB of the high product. The contribution of this rounding trick has not been fully tested.

Could the autocorrelation phase be partly responsible? Simulations show no overflows or saturations. However, since each r[i] is the sum of 256 truncated products, each r[i] is low by a random variable distributed equally from 0 to 255, with a mean error of 128. It would probably help to add 128 to every final sum. However, such errors should matter most for the small sums computed on fricatives, which our system handles well, and matter least on loud vowels, when we fail! We must still conclude that matrix inversion is the weak link, although better accuracy in distinguishing weak fricatives would no doubt result from double precision autocorrelation.

Differentiation of the input by the IOP will be tried soon; not only will this lower the failure rate of matrix inversions (by reducing the low-frequency energy which contributes to C≥1.0), but it will also compensate the overall -6 dB/octave vocal tract characteristic and show the 2nd and 3rd formants better [4].

Future Work. Top priority naturally goes to improving the accuracy of the system, as by using double precision on the matrix and perhaps the autocorrelation. The latter can be speeded up by a factor of at least 2 and probably 4; presently autocorrelation uses only one of the four AS multipliers. Reprogramming will reduce the Phase 1 time from 5 to 1.5 msec. True real-time operation will occur when the system becomes part of the Hearsay II implementation on C.mmp [5]. Even though double precision may be required throughout, we still believe that real-time estimation of vocal-tract resonances is possible on the SPS-41.

Acknowledgements.

The author is indebted to Quentin Klein of SPS, Inc., who is largely responsible for the autocorrelation and helped in early implementations of the matrix inversion. This research was sponsored by the Advanced Research Projects Agency of the Secretary of Defense (F44620-73-C-0074) and is monitored by the Air Force Office of Scientific Research.

References

[1] John D. Markel, Formant Trajectory Estimation from a Linear Least-Squares Inverse Filter Formulation. SCRL Monograph No. 7, Santa Barbara, California, 1971.

[2] N. Levinson, "The Wiener RMS Error Criterion in Filter Design and Prediction," Journal of Mathematics and Physics, Vol. 25, No. 4 (1947) pp. 261-278

[3] E. A. Robinson, Statistical Communication and Detection with Special Reference to Digital Data Processing of Radar and Seismic Signals, Hafner Publishing Co., New York, 1967.

[4] J. D. Markel and A.H. Gray, "On Autocorrelation Equations as Applied to Speech Analysis," IEEE Transactions on Audio and Electroacoustics, April 1973 pp. 69-79.

[5] Erman, et al, "The Hearsay II Speech Understanding System: Overview and Organization," Proceedings of the IEEE Symposium on Speech Understanding, 1974 (this volume).

A 16-BIT A-D-A CONVERSION SYSTEM FOR HIGH FIDELITY AUDIO RESEARCH

Stan Kriz

Computer Science Dept., Carnegie-Mellon University, Pittsburgh, Pa.

ABSTRACT

An A-D and D-A converter system with exceptionally wide dynamic range and low distortion is discussed. The converters include a special track and hold circuit which eliminates slewing distortion, active low pass tilters, and data buttering queue.

Introduction

Traditional 12-bit analog-digital-analog conversion ot high quality audio is becoming insutticient tor audio analysis and synthesis research. The need for greater dynamic range and low distortion has led to the development ot a 16-bit converter system at Carnegie-Mellon University designed specifically for audio service. The system has a total dynamic range of 90 dB., and less than 0.1 percent distortion and noise at large signal amplitudes. Conversion periods trom 20 microseconds to 150 microseconds are programmable and an appropriate low pass tiltor is selected automatically. Direct memory access to a minicomputer and a 64 word data queue provide simplified programming.

Conversion Technique

Figures 1 and 2 show schematically the operation of the DAC and ADC respectively. Rather than use tull 16-bit converters, the system first prescales the 16-bit digital (or analog) signal to form a quasi-tloating-point number. Twelve bits beginning with the first signiticant bit are taken as a floating-point "traction" while a 3-bit "exponent" signities the position, or magnitude, of the "fraction". Only the 12-bit "fraction" is converted and atterward the analog (or digital) signal is postscaled by the "exponent" to restore proper magnitude. This technique extends the dynamic range ot 12-bit conversion by 24 dB without incurring the expense and stability problems ot true 16-bit converters. As with conventional designs, track and hold circuits are employed on the DAC to deglitch the converter, and on the ADC to permit successive-approximation conversion.

Track and Hold

It is not generally recognized that the DAC track and hold can create considerable distortion. The usual transition behavior of commercial track and hold circuits consists ot a slew period followed by quick and exact settling to the new signal level. Because this transition slewing is not superposition linear, heterodyning ettects between the input signal and the sampling clock may occur. For example, two microseconds is a typical slewing time for a tull scale transition. If a maximum amplitude sinusoidial input signal ot 7 KHz is sampled at a rate ot 20 KHz, a 1 KHz heterodyne ot approximately -35 dB amplitude will be produced. In this case the input signal is sampled three times per cycle, and the resulting slewing assymetries repeat every seven cycles.

Changing the track and hold transition behavior to a simple exponential decay results in non-slewing transitions which maintain superposition linearity. Although long settling time makes exponential decay useless tor most commercial applications, audio signals incur only slight changes of amplitude and phase. The track and hold designed for the 16-bit converters has an exponential time constant of about 0.5 microsecond and the resulting slight high frequency roll-ott can be compensated by an external network.

As a turther modification for audio service, overall DC feedback may be used around the track and hold. Since the audio signal is sampled linearly and no heterodynes (including DC) are tormed, the output can be integrated and fed back to suppress any DC errors. The complete D-A system diagram in tigure 3 shows that the DC feedback loop includes all amplitiers to the output connector where an ottset of less than one-half LSB can be easily maintained. The complete A-D system pictured in figure 4 uses a digital integrator and a small DAC to maintain zero digital offset in the output data.

Low Pass Filters

Any audio conversion system clearly must have low pass filters commensurate with system quality. The trequency-dependent negative resistance (FDNR) active tilter contiguration permits the design ot component tolerant tilters with very low distortion and wide dynamic range [1]. Because varying audio requirements make the optimization of tilter parameters ditticult, the filters were built as easily moditiable modules. Any standard contiguration, ladder filter ot order nine or less may be implemented by changing a few resistors. Standard ninth order elliptic-tunction values (to 1 porcent tolerance) prosently are being used. Passband equals 87 percent of the Nyquist frequency with 0.5 dB measured ripple. The stopband attenuation at the Nyquist frequency and above measures greater than 68 dB, and signal to noise ratio (20 KHz bandwidth) exceeds 95 dB.

This research was supported by the Advanced Research Projects Agency ot the Ottice ot Defense under contract number F-44620-73-C-0074 and is monitored by the Air Force Ottice of Scientitic Research.

Several conversion rates are commonly required by the user community. To facilitate ease of operation, four low pass filters with differing cut-off frequencies are installed in both the A-D and D-A. A buffer amplifier is necessary at the output of each filter and includes a peaking network to compensate high frequency roll-off phenomena (including track and hold) which are a function of conversion rate [2]. A peak of about 6 dB is required, and the networks provide a few dB of additional stopband attenuation.

System Features

The systems are designed for convenient user operation. Figures 3 and 4 show that the A-D and D-A are independently interfaced to a PDP-11 minicomputer. Data, usually divided into large blocks, is transfered by direct memory access (DMA). Processor attention is not required except for interrupt service at the completion of each block transfer. During these interrupts, a 64 word first-in-first-out (FIFO) queue provides several milliseconds of buffering to permit continuous data flow without critical interrupt timing.

A crystal clock divider provides four program selectable conversion clocks between 20 microseconds and 150 microseconds. Programing the clock rate simultaneously connects the appropriate low pass filter from the set of four filters. Timeout circuitry clears the converters and FIFO's between user operations to eliminate annoying clicks at the startup and conclusion of conversion.

For monitoring purposes, the D-A has provision to echo the A-D output independently of the processor. All of the D-A inputs including clock and filter selection automatically switch to echo mode for the duration of A-D operation.

Performance Tests

The D-A system was tested by converting perfect digital sinewaves of varying amplitude and frequency. The fundamental sinewave was removed from the analog output of the converter system with a compensated twin-tee filter and the resulting residue (all noise, harmonic distortion, and heterodynes) is plotted in figure 5. For low amplitude signals, the random noise of the active filter and the DAC quantization noise are about 3 dB above the theoretical minimum quantization noise of the conversion. As the peak sinewave amplitude is increased above twelve bits, conversion noise rises because of the floating-point operation of the converter which truncates low order bits. In this region total residue is about 0.03 percent, and harmonic distortion becomes noticable only near maximum amplitudes. The increase in residue at 0 dB, 12 KHz input is a heterodyne caused by slight distortion in the active low pass filter.

The A-D system test was somewhat cumbersome but provides preliminary information until a through test can be implimented [3]. Similar to the D-A test, a low distortion sinewave (noise and distortion more than 80 dB down) was converted and the fundamental subtracted (digitally) from the output. The residue was then digitally amplified and reconverted to analog for examination. Figure 6 shows a noise shelf of about 90 dB : about 8 dB above the theoretical minimum. This higher level is partially attributable to track and hold sampling of high frequency noise from the filters. In general, quantizing noise at all input amplitudes is higher because of sensitivity of the analog circuits driving the 12-bit ADC. Both of these factors hopefully can be reduced in the near future.

Conclusion

A 16-bit A-D-A conversion system has been designed specifically for high fidelity audio service. The system utilizes floating-point approximation at conversion, a linear track and hold circuit, and overall DC feedback. Major user features include easily modified low pass filters, a DMA minicomputer interface, and a 64 word data buffer. System performance approaches theoretical limits.

Acknowledgements

The author would like to thank Paulo Coraluppi and Raj Reddy for their contributions to the system design, and Rich Lang and Mark Firley for their patient help with the hardware. Thanks also to Thomas Stockham Jr. at the University of Utah for several helpful conversations.

References

[1] L. T. Bruton, "Network transfer functions using the concept of frequency-dependent negative resistance", IEEE Transactions on Circuit Theory, vol. CT-16, pp. 406-408, August 1969.

[2] T. G. Stockham Jr., "A-D and D-A converters: their effect on digital audio fidelity", in 41st meeting of Audio Engineering Society , N.Y. C., Oct. 5-8, 1971.

[3] Chin-Moh Tsai, "A digital technique for testing A-D and D-A converters", M.S. thesis, University of Utah, June 1973.

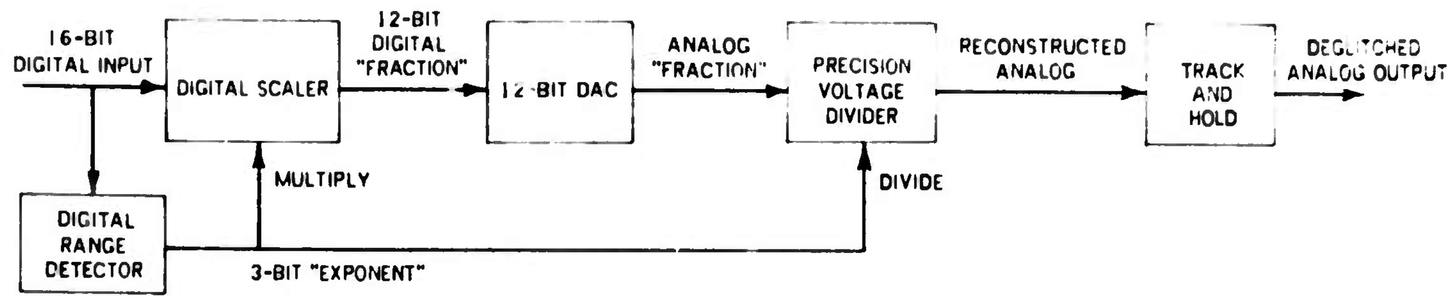


Figure 1. Operation of the 16-bit DAC.

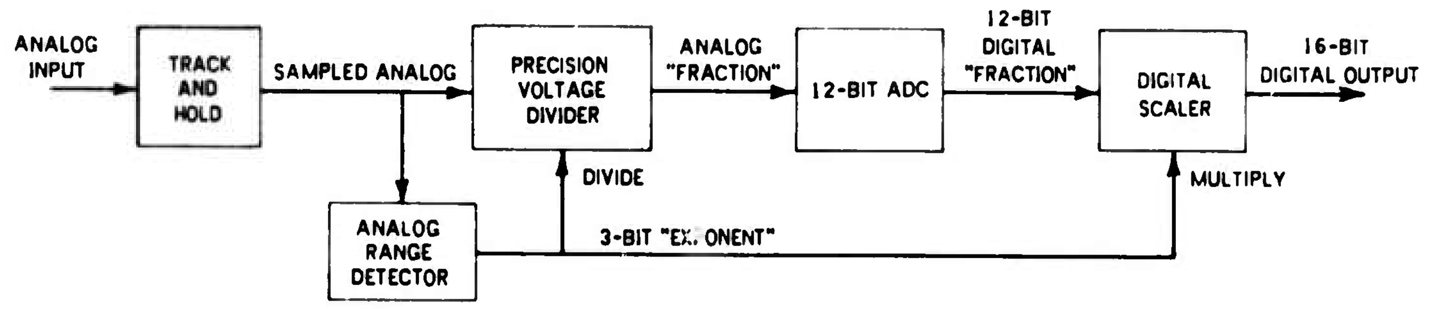


Figure 2. Operation of the 16-bit ADC.

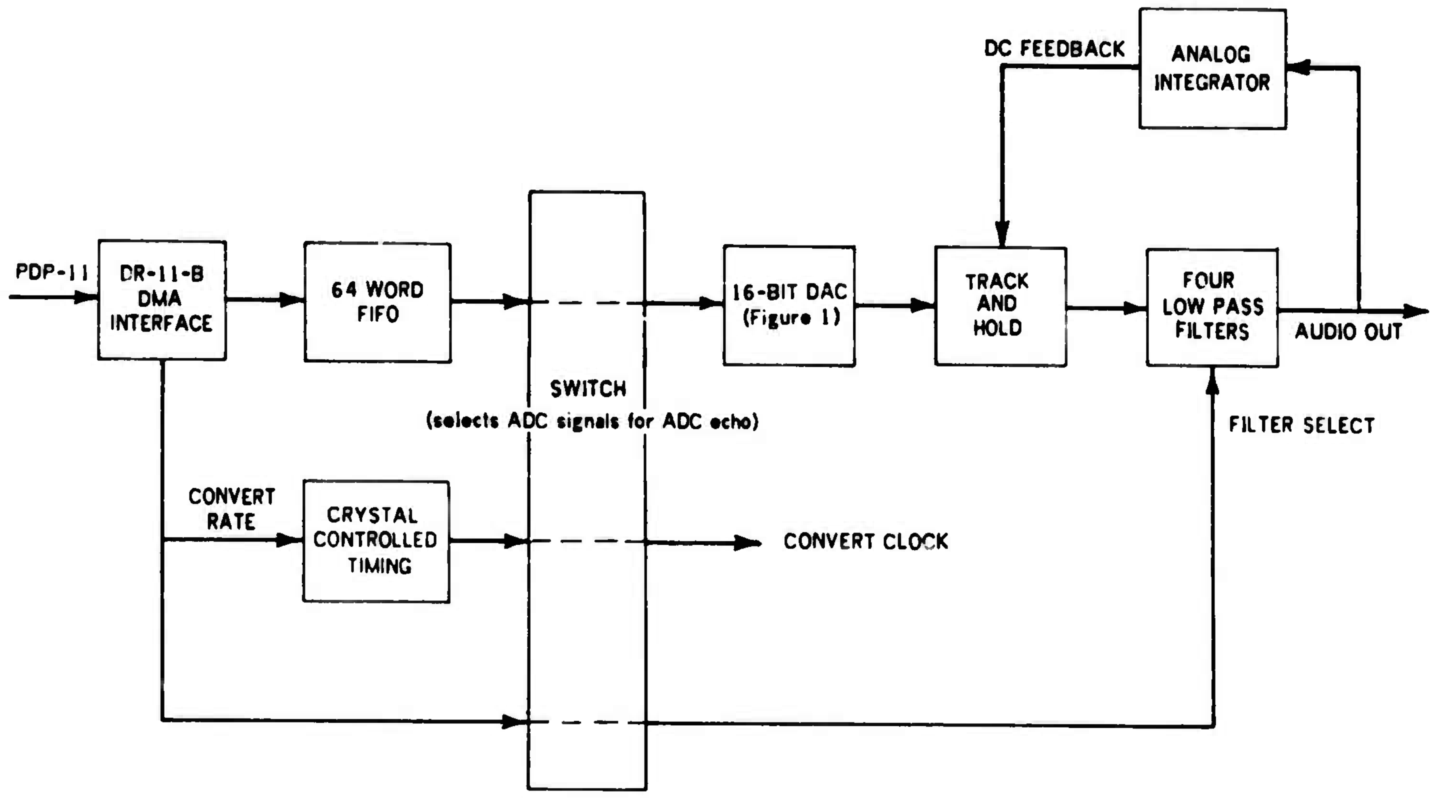


Figure 3. The complete D-A system.

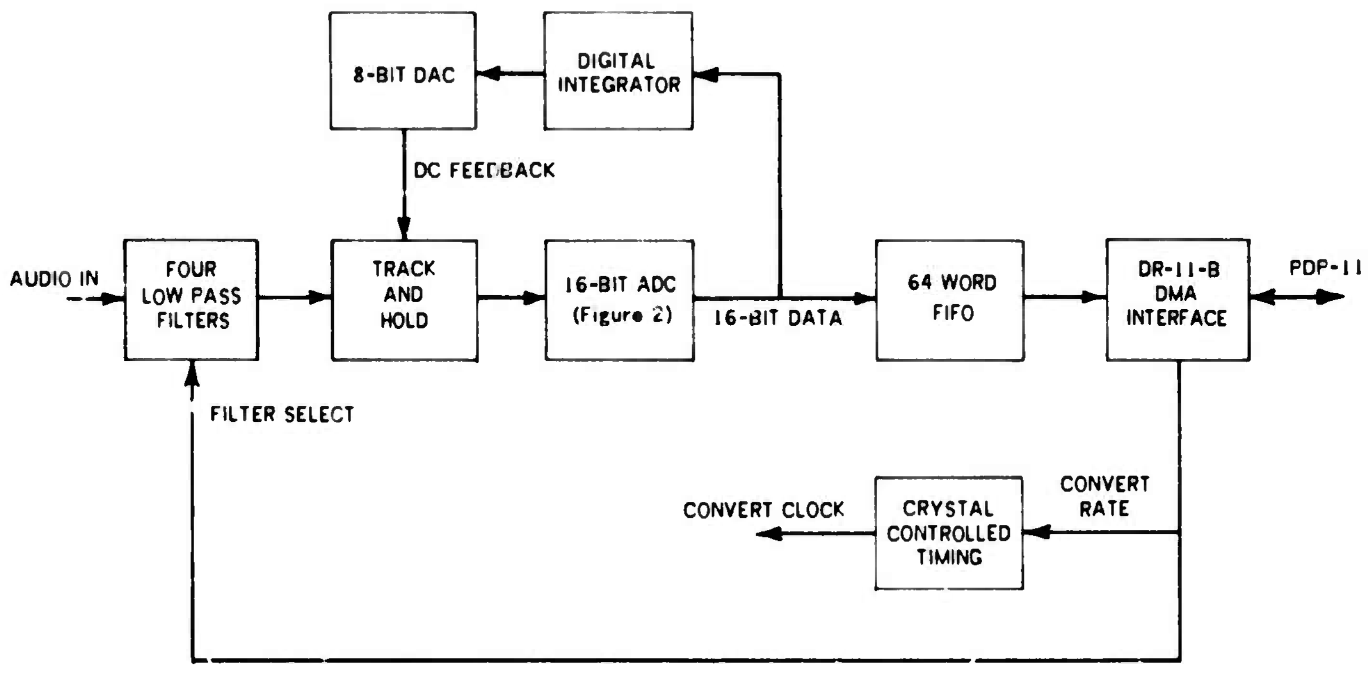


Figure 4. The complete A-D system.

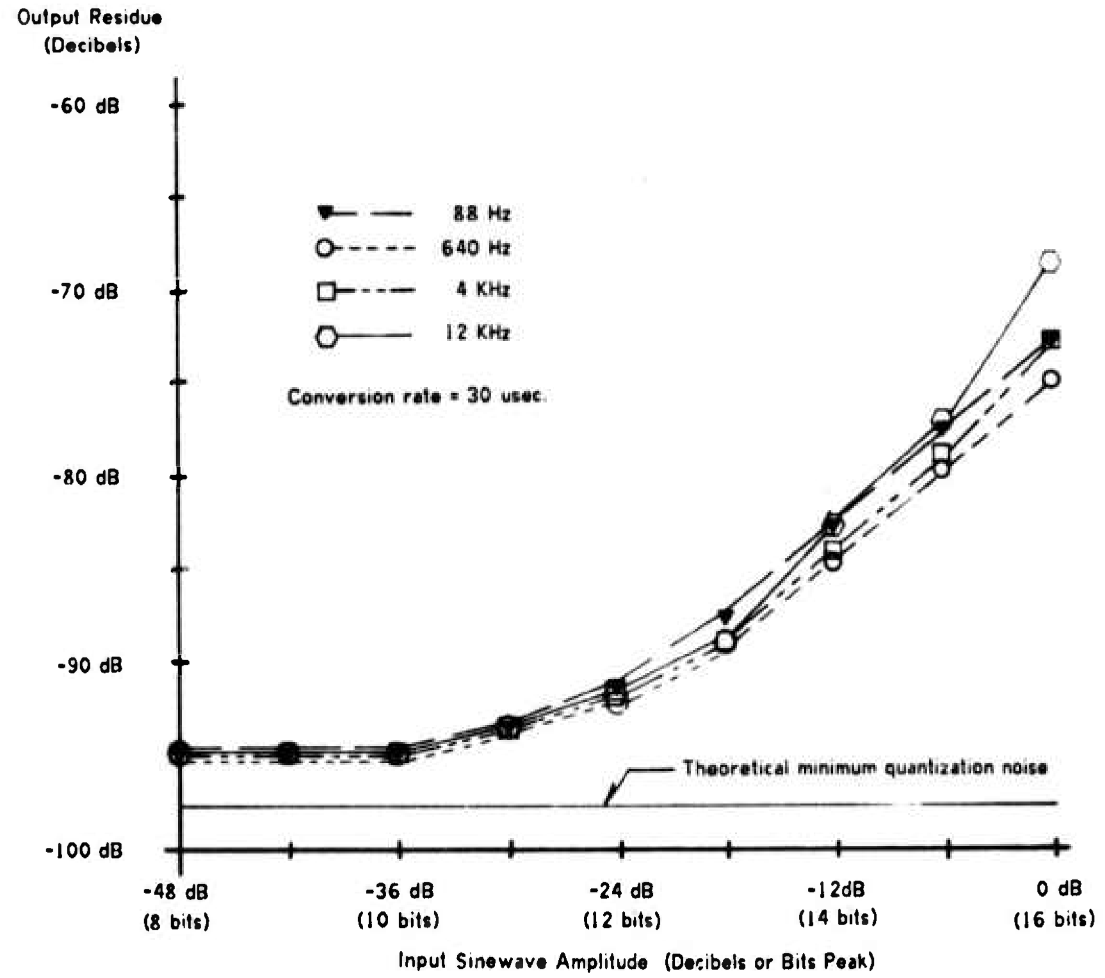


Figure 5. Performance of the D-A system.

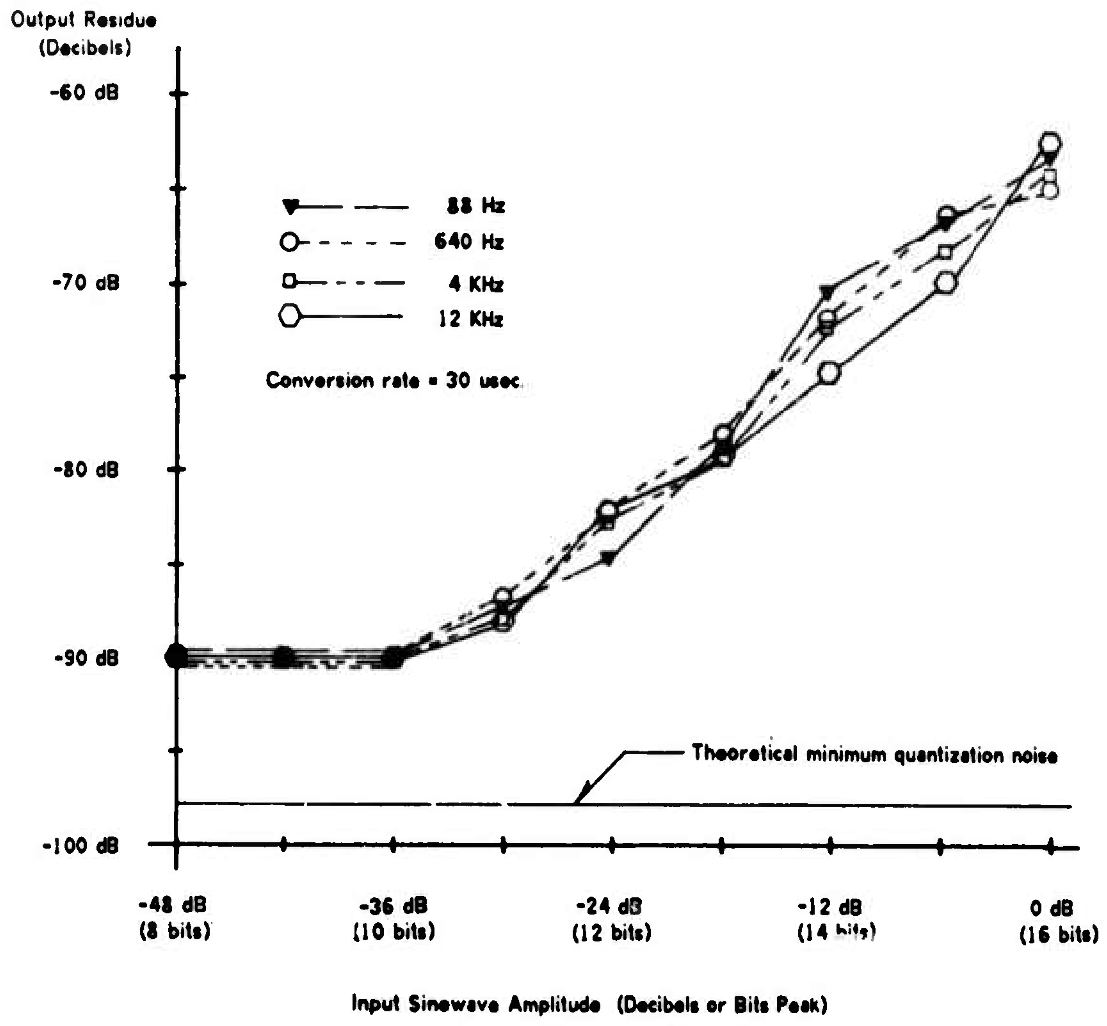


Figure 6. Performance of the A-D system.